# LEARNING SALIENCY FOR HUMAN ACTION RECOGNITION

Submitted in partial fulfillment of the requirements
of the Degree of Doctor of Philosophy

Daria Stefic
School of Electronic Engineering and Computer Science
Queen Mary, University of London

Supervisor: Dr. Ioannis Patras

June, 2016.

**Statement of originality**

I, Daria Stefic, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third partys copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.



Signature: Daria Stefic
Date: 11.06.2016

Details of collaboration and publications:

- D. Stefic, I. Patras. Action recognition using saliency learned from recorded human gaze. Under revision for Image and Vision Computing (IVC), Elsevier.
- D. Stefic, I. Patras. Learning visual saliency using topographic Independent Component Analysis. International Conference on Image Processing (ICIP) 2014.

**Abstract**

When we are looking at a visual stimuli, there are certain areas that stand out from the neighbouring areas and immediately grab our attention. A map that identifies such areas is called a visual saliency map. As humans can easily recognize actions when watching videos, having their saliency maps available might be beneficial for a fully automated action recognition system. In this thesis we look into ways of learning to predict the visual saliency and how to use the learned saliency for action recognition.

In the first phase, as opposed to the approaches that use manually designed features for saliency prediction, we propose few multilayer architectures for learning saliency features. First, we learn first layer features in a two layer architecture using an unsupervised learning algorithm. Second, we learn second layer features in a two layer architecture using a supervision from recorded human gaze fixations. Third, we use a deep architecture that learns features at all layers using only supervision from recorded human gaze fixations.

We show that the saliency prediction results we obtain are better than those obtained by approaches that use manually designed features. We also show that using a supervision on higher levels yields better saliency prediction results, i.e. the second approach outperforms the first, and the third outperforms the second.

In the second phase we focus on how saliency can be used to localize areas that will be used for action classification. In contrast to the manually designed action features, such as HOG/HOF, we learn the features using a fully supervised deep learning architecture. We show that our features in combination with the predicted saliency (from the first phase) outperform manually designed features. We further develop an SVM framework that uses the predicted saliency and learned action features to both localize (in terms of bounding boxes) and classify the actions. We use saliency prediction as an additional cost in the SVM training and testing procedure when inferring the bounding box locations. We show that the approach in which saliency cost is added yields better action recognition results than the approach in which the cost is not added. The improvement is larger when the cost is added both in training and testing, rather than just in testing.

**Acknowledgements**

The first and the biggest thank you goes to Yiannis, for his immense patience and positive attitude, but especially for all his negative comments. Second thank you goes to Petar, the person who was most present during my PhD time, as a flatmate, colleague, travel companion, and a friend (still, I hope). I am thankful to other lab colleagues for making a nice work atmosphere: Oya, Saverio, Stefano, Vlado, Fiona, Chris, Sertan, Vijay, Julie, Vangelis, Ivan, Yoshiki. I am also thankful to those who were not around physically but kept my spirits from the distance and helped me to recover during my home visits: my family and friends, especially Petra, Hana, Suzana, Manuela, Jelena, Mateja, Antonio, Dragana. Special thank you goes to Dora - this adventure would have never started if it were not for her initial motivation. And that would have been a shame.

# Contents

**List of abbreviations**

| | |
|---|---|
| BoW | Bag of Words |
| CNN | Convolutional Neural Network |
| CRF | Conditional Random Field |
| FV | Fisher Vector |
| HMM | Hidden Markov Model |
| HOF | Histogram of Optical Flow |
| HOG | Histogram of Oriented Gradients |
| HOMB | Histogram of the Oriented edges of the Motion Boundaries |
| ICA | Independent Component Analysis |
| ISA | Independent Subspace Analysis |
| LDA | Latent Dirichlet Allocation |
| LSTM-RNN | Long Short Term Memory Recurrent Neural Network |
| PCA | Principal Component Analysis |
| pLSA | probabilistic Latent Semantic Analysis |
| RNN | Recurrent Neural Network |
| SIFT | Scale-Invariant Feature Transform |
| STIP | Spatiotemporal Interest Point |
| SVM | Support Vector Machine |
| tICA | topographic Independent Component Analysis |

# List of Figures

# List of Tables

# 1 Introduction

## 1.1 Motivation and objectives

Human action recognition has been a well-studied problem in the last decade. It has many potential applications including automated surveillance, video archival/retrieval, medical diagnosis, sport analysis, and human-computer interaction. However, this is a complex and challenging task due to many ambiguities that are caused by non-rigid body articulation, loose clothing, occlusions, as well as by image/video noise, shadows, viewpoint/scale/illumination changes, etc. Designing features that are invariant to all variations that occur in the data is a very challenging problem. Another problem is *where* to look for the relevant features. Videos that contain actions are commonly unsegmented and it may be required to annotate certain frames in terms of, for example, bounding boxes that would provide locations of the actions. This is a very cumbersome and time-consuming process, so we would want our action recognition system to be able to do that automatically, without human intervention. For that reason, human action recognition is usually posed as a weakly labeled problem, i.e. a problem of action clip classification in which only the label of each action clip is known, but the labels of individual parts of the action clip are not. This means that semantic labels available for a video refer to the class of the video as a whole, whereas the discriminative features that result in that categorization may only occur over a spatiotemporal subset of the video. Therefore, a fully automated action recognition system should be able to simultaneously learn discriminative features and find parts of a video where those features occur.

Using gaze information as a form of an input can enable a computer vision system to gain more information about a given task, such as object detection/recognition, image segmentation and action detection/recognition. Analyzing the way humans look at images/videos could lead to improvements in the construction of computer vision systems because, if class specific observation patterns exist, decisions regarding the computational recognition and/or detection process could be made based on human gaze data[10]. For example, when we observe a video of two people shaking hands, perhaps we would move our eyes to fixate on areas that we deem important: peoples

faces and their hands. If there was a video of a couple of people playing football, we would probably fixate on their lower body parts and/or a ball. Even though in both of those videos people (who may look very similar) are performing an action we want to recognize, we fixate our eyes on different areas, depending on the action class of the video. The body parts that we fixate, that is hand in handshaking and leg in kicking, are the ones that are relevant to discriminate between the actions. Therefore, by focusing only on the areas where people have looked, for a fully automated action recognition system it would be easier to discriminate between those two videos as it would not have to look for those areas. Indeed, using human fixations has shown to be useful in many real world applications, such as image [11, 12] and video retrieval [13], image segmentation [14], assistive robotics [15], object detection/recognition[16, 17] and action recognition as well[18, 19].

However, if the gaze information about fixated areas is not available, a fully automated action recognition system would have to infer those areas and patterns automatically. This makes the learning more challenging. Having available the information about the relevant areas, it might be that the recognition system would be easier to train, especially given the limited amount of training data. What we aim is to build a machine learning framework for the problem of action recognition that would be able to use gaze information in order to alleviate this problem. Since eye tracking provides a non-invasive way of obtaining user information, and as the devices continue to reduce in size and cost, they may become one of the most informative and natural sensor mechanisms for gathering useful user data. However, we would not want to use eye-tracking for each new test video. Hence, in a test video we would want to be able to predict the location where the relevant parts of the action occur, while using ground truth eye-tracking data only during training.

Another important goal of eye-tracking studies is understanding the human visual system and the visual process. Understanding the mechanism of human visual attention is a problem that has attracted the interest of several research areas, such as cognitive science, neuroscience, biology, psychology and computer science. For example, building computational models of (visual) cortex by drawing inspiration from biology is very common in the area of computer science and computer vision in par-

ticular. [1] One of the tasks of such computational models is human fixation/saliency prediction, which is a problem that has been well studied in the recent years within the computer vision community.

Visual saliency is usually defined as the distinct subjective perceptual quality which makes some items in the world stand out from their neighbors and immediately grab our attention [20]. The basic principle behind computing saliency is the detection of locations whose local visual attributes significantly differ from the surrounding image attributes, along some dimension or combination of dimensions. The difference could be in a number of simple visual feature dimensions which are believed to be represented in the early stages of cortical visual processing: color, edge orientation, luminance, or motion direction [21]. For more details on which elementary visual features may strongly contribute to saliency prediction see [22].

It is important to note that visual saliency is subjective, i.e. it is the consequence of an interaction of a stimulus with other stimuli, as well as with a visual system. Therefore, it cannot be described as a physical property of a visual stimulus. Also, what is salient for humans may not be salient for birds. What is salient can be influenced by training: for example, for human subjects particular letters can become salient by training. In our work we will focus on the saliency of human subjects and we will aim to reconstruct their saliency maps. The assumption we are making here is that humans are good at localizing and recognizing actions in videos and therefore their saliency maps are informative for the task of action recognition.

Until recently, most computational approaches that are trying to model the prediction of human fixations/saliency were based on manually selected features chosen based on some assumptions that we have about our visual system. Such approaches are limited by the knowledge of human visual system. Rather than manually designing the features for saliency prediction we aim to learn them.

This concept of learning features has come to the focus of computer vision community in the last few years, even though basic ideas have been around for a very long time. For example, many of commonly used existing linear models, such as PCA and ICA are actually one layer architectures in which the coefficients are learned

[1] Although, building biological and computational models is a bidirectional process: in return, analysis of computational models combined with eye tracking studies could hopefully shed some light on the biology of the human visual system.

in an unsupervised manner. The idea of learning the features (in a supervised or an unsupervised manner) is the first main characteristic of deep learning paradigm which has become very popular lately. Second main characteristic is stacking the features in a hierarchical way in order to make deeper architectures with multiple feature layers of increasing complexity. This characteristic is inspired by the human brain architecture which is know to be organized in a hierarchical way. The intent is to discover more abstract features in the higher levels, which will hopefully make the classification task easier. As mentioned previously, manually designing features for either saliency prediction or action recognition is a challenging problem. Instead, we aim to employ this idea of learning features for both of those problems.

## 1.2 Contributions

The main goal of this thesis is to address the problem of action recognition using saliency *learned* from recorded human gaze information.

In the first phase the focus is on developing a saliency prediction framework that learns saliency features at multiple layers and uses gaze information during supervised learning of features and a saliency classifier. We have addressed this problem both for images (2D) and action videos (3D). In the context of the third chapter, we refer to the first layer features as low-layer features, to the second layer features as mid-layer features, and to the features higher than second layer as high level features. Our contributions regarding the problem of 2D and 3D saliency prediction are the following:

1. we have employed a biologically inspired unsupervised learning algorithm, namely topographic ICA, in order to learn low-level 2D saliency features. Saliency learning is posed as a binary classification problem where gaze fixations are used as a ground truth supervision data and linear SVM is used as a classifier on top of learned saliency features. Note that the supervision from gaze is used only in the SVM learning. We have shown that our learned features outperform handcrafted ones.

2. in the same topographic ICA + SVM framework we have incorporated supervised learning from gaze in mid layers of topographic ICA and shown that this

approach outperforms the one in which supervised learning is not incorporated in mid layers of topographic ICA, i.e. in which supervised learning is incorporated only at the highest level, that is SVM learning (as in 1),

3. we have employed a fully supervised multilayer 2D convolutional neural network that uses only gaze fixation data as a ground truth supervision at all layers and shown that this type of network outperforms our previous approaches,

4. we have extended 2D convolutions to 3D and shown that the results of a convolutional neural network that uses 3D convolutions outperform the results of the network that uses 2D convolutions.

In the second phase the focus is on developing methods that use learned saliency prediction in order to find parts of videos that contain action and by doing so alleviate the action classification. Our contributions regarding action recognition using saliency prediction are the following:

1. we have learned 3D local action features using gaze information when sampling cuboids for the learning. We have shown good discriminatory power of those features by comparing them to the commonly used handcrafted ones in a simple majority voting framework. In the same framework we have shown the importance of using saliency prediction when sampling cuboid votes,

2. we have built a latent SVM based action recognition framework that uses learned action features and learned saliency in order to infer latent variables, i.e. bounding boxes of the actions. We have shown the improvement over baseline SVM that does not use bounding box inference. The improvement is larger when bounding box inference is used during both training and testing, then when it is used only during testing. Finally, we have shown that using saliency prediction significantly improves the results compared to the same SVM framework in which saliency is not used.

In Fig. 1 we illustrate the overview of the work that will be presented in the thesis and highlight in which part of the presented work each of the contribution will be shown.

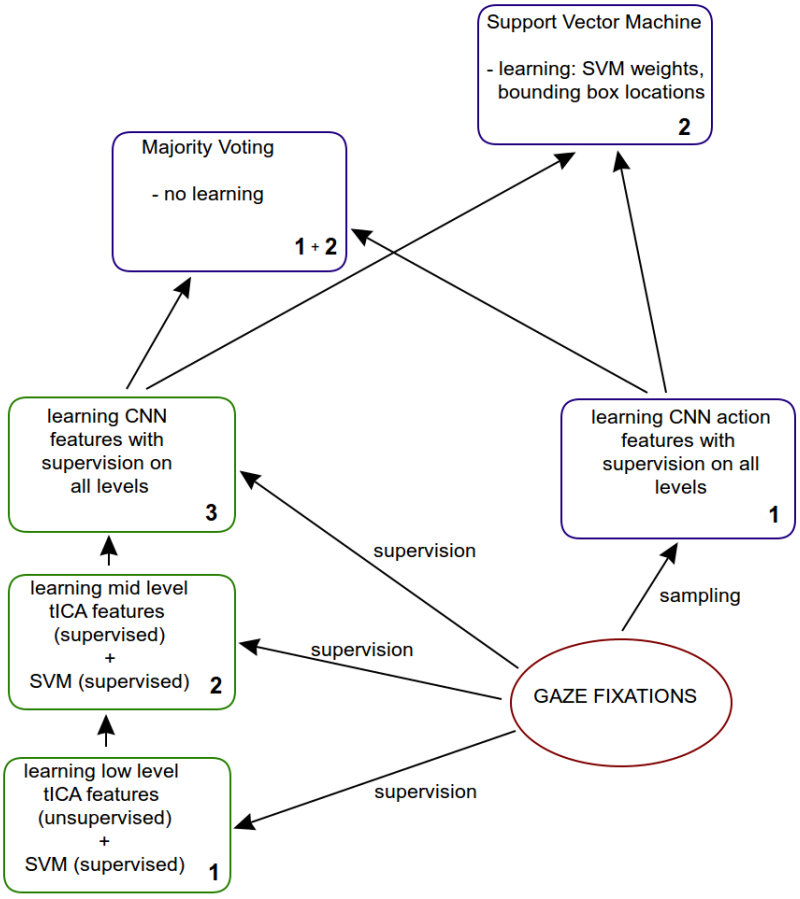



Figure 1: Thesis overview. Green components represent the work related to saliency learning (Chapter 3) and blue components represent the work related to the action recognition (Chapter 4). Numbers in the bottom right corners are related to the list of the contributions, separately for saliency learning and action recognition.

## 1.3 Outline

The rest of the thesis is organized as follows.

In chapter 2 we present related works: in 2.1 we present works related to visual saliency learning, and in 2.2 we present works related to human action recognition.

In chapter 3 we present our multilayer architectures for predicting fixations that uses learned features and recorded human fixations. We show how these features compare to the commonly used handcrafted features and how learning features while

incorporating supervision from human gaze at different layers affects the results.

In chapter 4 we present two methods for action recognition that use saliency learned from recorded gaze fixations. In 4.1 we present a simple majority voting framework in order to demonstrate the discriminatory power of our learned action features. In 4.2 we build upon those features a latent SVM framework that uses additional cost from saliency prediction and demonstrate the importance of incorporating this loss in the SVM.

In chapter 5 we present the conclusions of our work and propose some future research directions.

# 2 Related work

## 2.1 Visual saliency learning

In the last decade, many computational models for predicting human saliency have been proposed. They are based on various, usually biologically inspired, principles, such as locally occurring patterns[23, 24], information maximization principle[25, 26], graph representations[27], filtering methods[28] and most recently neural networks[29, 30, 31, 32]. Typically, saliency maps obtained by recording human gaze are used only to validate the proposed models. One of the pioneering and most well known works that validates their saliency prediction model in such way is the one from Itti *et al.* [23]. Their saliency prediction model is based on computing three types of feature maps: intensity-, color-, and orientation-based. In the next processing step, a map normalization operator is proposed. This operator promotes maps in which a small number of strong peaks of activity is present, while suppressing maps that contain numerous comparable peak responses. In this model no learning at any stage is employed, either supervised (using gaze fixations as a ground truth) or unsupervised.

More recent approaches in predicting visual saliency use recorded eye gaze information in order to train classifiers that predict human fixations [33, 34, 35, 36, 2, 37, 35]. A detailed review on recent advances on learning saliency using recorded eye gaze information in the training is given in [38]. The earliest work following this approach is the one of [33, 34, 35]. They extract fixated and non fixated patches as positive and negative examples, respectively, and train a binary RBF SVM classifier on those raw patches. Note that no manually selected features are used: an SVM classifier is trained on raw image patches. On their dataset this approach yields superior results in predicting human fixations comparing to the model of Itti *et al.* [23].

In contrast to this, [36, 37, 2] use manually selected features. They use the same type of low level features as [23], but in addition to those, [37, 2, 36] use high level features, such as the output of a face detector and a center bias, and [36] use mid level features, such as the output of a horizon detector, too. Face detector output is an important feature. In [39] is found that when faces are present in an image, we first focus on them. Also, it is found that inter-subject scanpath consistency on images with faces is higher that in the images without them. Another important high

level feature is central bias. Most of datasets exhibit central bias in two ways. First, photos are usually taken in such way that objects of interest are centered. Second, when looking at images we naturally tend to look at the image center first. One way of modeling central bias is described in [37]. In this work central bias is used as a high level feature for saliency prediction. It is also used in the same way in the follow-up work from the same authors, that is in [2].

Central idea in the works that use combinations of manually selected low, mid and high level features [36, 37, 2] is to learn optimal weights for feature integration using recorded gaze fixations. In [37] optimal weights are learned using linear, least square regression. In a more recent work from the same authors, [2], optimal weights (for same feature channels) are learned using nonlinear AdaBoost. This yielded major improvements in the saliency prediction results over linear, least square regression. In [36] optimal weights are learned using binary linear SVM. Even though they use more types of features, their results do not outperform the ones from [2].

More detailed illustration of the bottom-up saliency model used in [2] (and similar to the models in [37, 36, 23]) is shown in Fig. 2. For an image location $\mathbf{x}$ the values of the maps $f_1, f_2, ..., f_n$ (based on color, intensity, orientation and face detection) at this particular location are extracted and stacked to form the sample vector:

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x})\ f_2(\mathbf{x})...\ f_n(\mathbf{x})]. \tag{1}$$

In the works of [37, 2, 36] this vector is assigned with a label +1 or -1, indicating whether the location is fixated or not, and then used for training a classifier that learns the optimal weights for features integration, while in [23] feature integration is handcrafted. When testing, $G(\mathbf{x})$ is the predicted value of saliency at location $\mathbf{x}$.

**Spatiotemporal saliency**

As in the case of saliency prediction in static images, there are many computational models for predicting saliency in videos. An exhaustive overview and comparative study is given in [40]. However, there are not many works that use human fixations in the learning phase, which is the focus of our work. To the best of our knowledge, there are only four works that are using recorded human fixations ([41, 42, 3, 43])

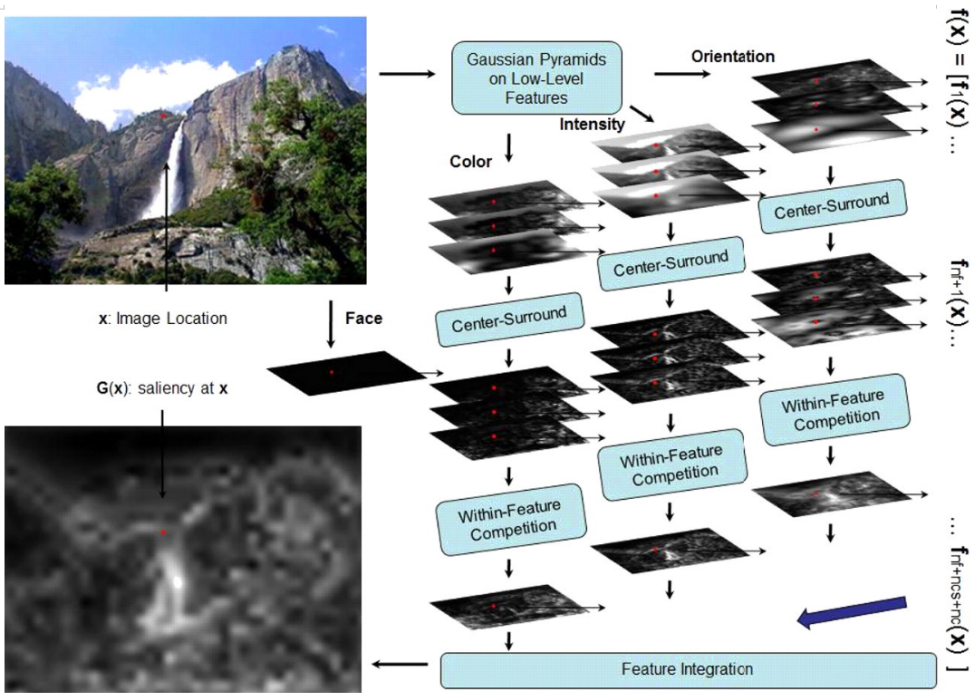



Figure 2: Bottom-up saliency model used in [2]

when learning the parameters for saliency prediction. Two of them, [43] and [3], use action videos datasets to validate their method, both in terms of saliency prediction and action classification results.

The first work that introduces the concept of using human ground truth fixations in the spatiotemporal saliency learning is the one of [3] (the same authors introduced this concept a year earlier for static images in [33]). In the same work they introduce the idea of using learned spatiotemporal saliency prediction for action recognition. Their spatiotemporal saliency predictor is based on the popular periodic detector [44]. In the work of [44] 1-D Gabor filters are used for spatiotemporal interest point detection. In contrast to this, in [3] temporal filters are learned using a simple neural network. In that way, the periodic detector is generalized to an arbitrary shaped temporal filters which are fitted to the eye tracking data by learning the network parameters, i.e. filter weights. This approach is validated in two ways. First, is is validated by showing that saliency prediction results of the learned saliency predictor outperform the saliency prediction results of the periodic detector and 3D-Harris

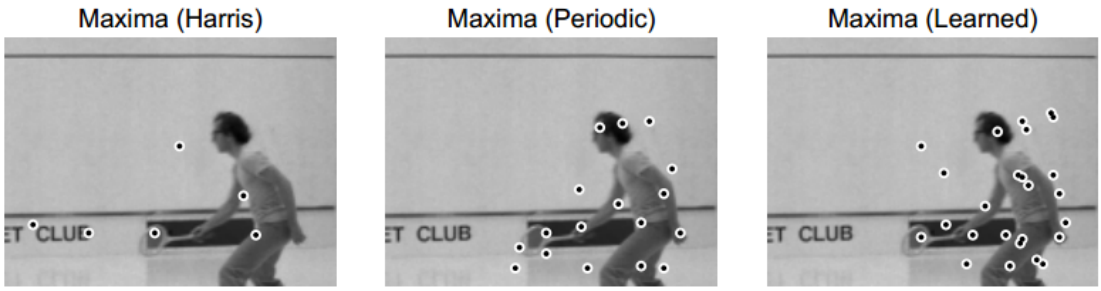



Figure 3: Illustration of detected maxima for Harris, periodic and learned detector [3]

detector, which are both completely manually designed, i.e. in which no parameters are learned. Second, it is validated by showing that action recognition results when using the proposed saliency prediction outperform the state-of-the-art which use either periodic or 3D-Harris detectors. Qualitative comparison of 3D Harris[45], periodic[44] and the proposed learned detectors responses is shown in Fig. 3. We can see that 3D Harris detector produces a response only if the spatiotemporal gradient varies significantly over all three dimensions; it detects *cornerness*. As opposed to this periodic detector and learned detector produce denser responses. We should also note that neural network is trained on short movie video clips and not on the action videos on which it is tested.

Another more recent work that addresses the problem of spatiotemporal saliency learning is the one of [43]. However, in contrast to [3] they collect human eye-tracking annotations for a couple of action datasets (Hollywood2 and UCF Sports) and use those for training their saliency predictor. They pose saliency prediction problem as a binary classification problem, i.e. fixated points are treated as positive examples, and non-fixated points are treated as negative examples. To solve this classification problem, a linear SVM is used on top of manually selected features extracted around points in question to learn the optimal weights for feature integration. This SVM then acts as a saliency predictor on test videos. This kind of learning concept was introduced for images in [36] and described previously; see Fig. 2. This work will be further described in 2.2.3: there we will focus on its contribution regarding action recognition.

The latest work that uses human gaze when learning saliency is [42]. In contrast to the majority of models that calculate saliency value for every pixel, in their model saliency is calculated only for a small set of candidate locations that are selected based on static, motion and semantic features. After candidate extraction, selecting the most salient ones is accomplished by learning the transitional probability, that is the probability to shift from one gaze location in a source frame to the new one in a destination frame. The learning problem is posed as a classification problem - whether a gaze transition occurs from a given source candidate to a given target candidate. It is shown that using sparse candidates achieves superior saliency prediction results and that it is fast.

To summarize, the common approach in saliency prediction was to use handcrafted local image features at the lowest layers of saliency prediction models and build a fusion layer on top of it. The fusion layer may or may not have been learned using recorded gaze information. We have presented the evidence that support the idea that recorded gaze, when incorporated in the saliency prediction models in the learning of the fusion layer improves the results comparing to the methods that do not learn the fusion from human gaze. We aim to see if learning at different layers of the architecture we will propose improves the saliency prediction results. We will do so by learning certain layers of our architecture in either unsupervised way or in a supervised way using human gaze as a ground truth.

In 2.1.1 we will present the work regarding topographic independent component analysis, an algorithm which can be seen as a layer-wise network and whose parameters are learned in an unsupervised way. This is the algorithm we will use in order to learn low layer features for saliency prediction. In 2.1.2 we will present the work regarding deep learning, where multiple layers of features can be learned in a supervised or unsupervised way, and show how this methodology has been used for learning saliency. In 2.1.3 we will present datasets we will use in our experiments.

### 2.1.1 Topographic Independent Component Analysis

Some works support the idea that applying Independent Component Analysis on natural images leads to the emergence of oriented linear filters that resemble simple-

cell receptive fields in the human visual V1 cortex [46, 47]. In order to explain the emergence of the V1 topography, i.e. complex cells that receive an input from simple cells, a more generalized model, namely topographic Independent Component Analysis (tICA), is presented in [46] . A detailed comparison between ICA and tICA is given in [47]. In that work it has been shown that tICA leads to simultaneous emergence of complex cell properties as well as to their topographic organization. The main idea in the tICA model comparing to ICA is to introduce nonlinearity in higher levels while considering the topographic ordering of basis vectors. This means that vectors with stronger higher order correlations should be close to each other in the topology. This kind of topological ordering might lead to the emergence of complex cell properties where each neighborhood cell acts like a complex cell[46].

Another algorithm very close to tICA is Independent Subspace Analysis (ISA), proposed by the same authors and in more details described in [8]. This model introduces nonlinearities as tICA, but the weights of the second layer of the proposed architecture are such that represent the subspace structure of the neurons in the first layer, instead of enforcing the proximity of vectors with stronger higher order correlations in the topology, as in tICA. In [7] ISA algorithm is used to learn spatiotemporal features for action recognition. Therefore, this work will be described in more details in 2.2.1.

Another unsupervised learning method that is closely related to component analysis methods is sparse coding. Both component analysis and sparse coding methods are based on generative modeling where an image or an image patch is represented as a linear combination of basis vectors multiplied by corresponding coefficients. Those coefficients vary from image to image, i.e. an image is characterized by those coefficiens. The main goal is to learn a set of basis vectors that will form a fixed dictionary and will be used to reconstruct new images or image patches. Component analysis and sparse coding methods can also be compared to wavelet transforms[48]: there, a manually designed dictionary is used to reconstruct images.

Independent Component Analysis and Principal Component Analysis are two commonly used component analysis methods. In PCA the goal is to learn a set of mutually orthogonal basis vectors that catupre the direction of maximum varince in the data and for which the coefficients are pairwise decorrelated. In ICA the goal is to learn a set of basis vectors that generate statistically independent coefficients.

The goal in sparse coding is to seek basis vectors by which each image is represented by a small number of active coefficients: sparse coding, unlike ICA, emphasizes sparsity over independence of the coefficients. However, a learning that maximizes the sparseness of the individual outputs also promotes their statistical independence[49].

Sparse image representations have been proposed as models of receptive fields in different parts of our visual system [49, 50, 5]. Most famous works that supports the idea that sparse image representations are a good way of modeling some parts of the human visual system is [49]. The search for sparse image code is formulated as a simple optimization problem in which a function that consist of two terms is minimized. First term is the reconstruction error which measures how well the code represents the images. Second term enforces the sparsity of the coefficients, i.e. it enforces the images being represented by only few active coefficients. The learned basis vectors are shown to have the same properties as the receptive fields of simple cells in mammalian primary visual cortex: they are spatially localized, oriented and bandpass.

Sparse image representations are commonly used to extract image features for the task of saliency prediction [51, 52, 31, 32]. In [52] two types of dictionaries are learned jointly. The first dictionary is learned in a sparse way and it aims to reconstruct few types of manually designed patch features. The second dictionary uses the sparse coefficients reconstructed using the first dictionary in order to reconstruct the patch fixation value. In [51] a discriminative dictionary for saliency detection is learned. The dictionary is learned in a sparse manner, but during the learning it also uses supervised information from eye tracking recordings in order to enhance the discriminative power of the dictionary. The works of [31, 32] employ sparse learning in a deep hierarchical manner and therefore will be described in section 2.1.2.

ICA has also been used for saliency prediction[53, 25, 54]. In those works ICA is used as a feature extraction method at lowest layer. As higher level modeling [53] use Bernoulli mixture model, [25] use information maximization principle and [54] use Bayesian framework. As mentioned, a very popular type of hierarchical non-linear modeling nowadays is by using deep learning models. Models that use deep learning for saliency prediction will be described in more detail in 2.1.2.

The algorithms mentioned above (ICA, tICA, ISA, sparse coding) have an unsupervised criterion according to which the weights of their architectures are optimized. However, there is an interesting work of [55] that shows how convolutional pooling architectures can be inherently frequency selective and translation invariant: good features can be extracted even with random weights. In support of this claim is also the work of [56]. Here, a standard convolutional architecture is used, and results obtained by an architecture with random weights at lower layers are shown to be comparable to the ones obtained by an optimized architecture. Here, by optimized architecture it is meant that an architecture is optimized using a supervision criterion, as opposed to [55] where an unsupervised tICA criterion is used.

On the other hand, there are many works that emphasize the importance of unsupervised pretraining and weight initialization of network weights [57, 58, 59, 60]. Also, there is a work of [61] that claims that when there is a large amount of labeled data available, there can be significant gain from optimizing the network weights using a supervised criterion comparing to the approach where an architecture is optimized using an unsupervised criterion.

### 2.1.2 Feature learning for saliency prediction

In the recent years there has been an increasing amount of work based on deep learning methodology. It has shown superior results over classical methods in various areas, including visual tasks, such as image classification[62], object detection [63], action recognition[64], audio tasks [65] and natural language processing [66]. The goal of deep learning is to learn a hierarchy of features. We can draw an analogy to the common pipeline in the vast majority of computer vision works[67]. In the first stage a feature descriptor is applied (for example: HOG, HOF, SIFT). In the second stage filters are learned in an unsupervised manner using K-means (the nonlinearity in this case is winner-takes-it-all). In third stage a pooling operator is applied as an average over multiple scales. Finally, the classifier is usually an SVM. Usually, each of this stage/layer is learned separately; in contrast to that, deep learning approach aims to learn all layers of the architecture end-to-end, i.e. learn all layers of the architecture simultaneously. However, what lacks in the vast majority of deep learning works is a mathematical analysis of network properties. There is an interesting work of

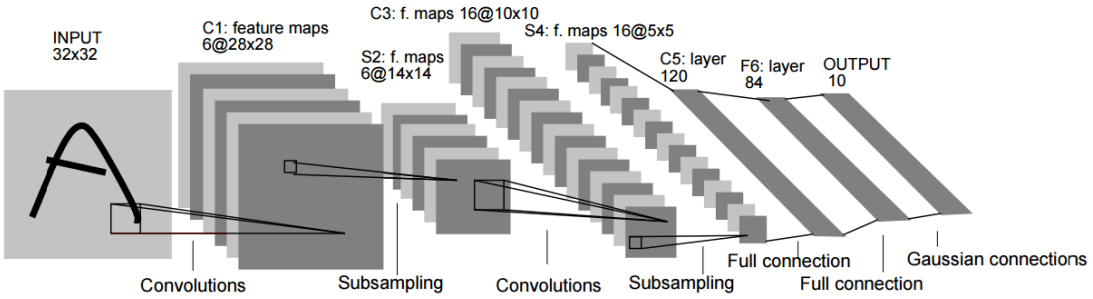



Figure 4: Convolutional neural network used in [4] for handwritten digits recognition

[68] which studies the properties of neural networks by concentrating on a particular class of deep convolutional networks defined by the scattering transforms. A scattering transform computes a translation invariant representation by cascading wavelet transforms and pooling operators, which average the amplitude of iterated wavelet coefficients. The first layer outputs SIFT-type descriptors, whereas the next layers provides a complementary invariant information for the given classification task. Using such transforms network properties can be analyzed mathematically in terms of the stability of the wavelet coefficients.

A very important concept in deep learning, besides the hierarchical learning of features, is the convolution. Except for being biologically plausible, convolution allows training of deeper architectures, as convolutional weight sharing reduces the number of parameters. One of the first and most famous convolutional neural network, LeNet-5, is presented in [69]. It is used for the task of handwriting recognition (see Fig. 4). This kind of, or similar, architecture is used in many different tasks, such as images, speech and time-series recognition[65, 70].

The selectivity of neurons for oriented bar stimuli in cortical area V1 is well documented and, as we have seen in 2.1.1, there are works that are trying to replicate the simple and complex cells of the V1 area. However, the response properties of neurons in the higher levels are still not well known. An attempt to model specifically the V2 area is made in [5] by using a sparse deep belief net whose layers are restricted Boltzmann machines. They show that using this method it is possible to capture a

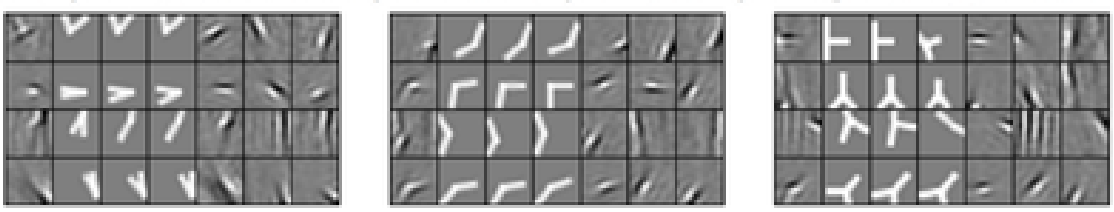


Figure 5: Visualization of V2 neurons in [5]

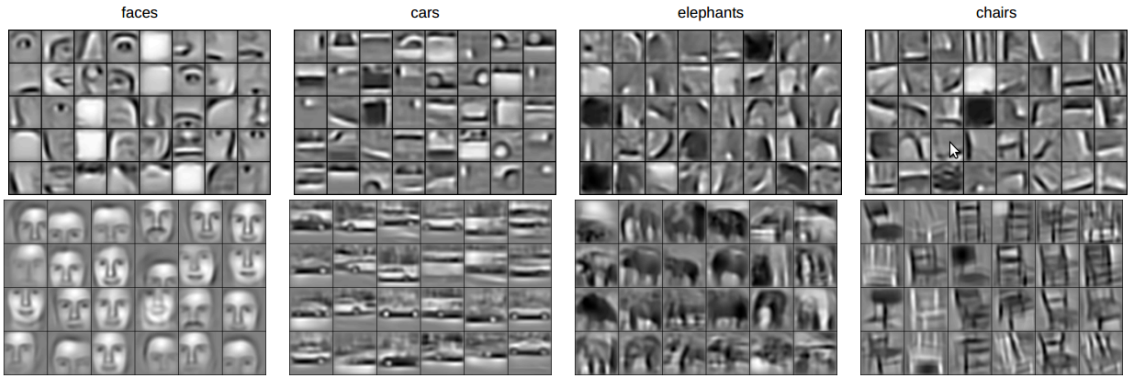



Figure 6: Filters learned by a convolutional deep belief network, as presented in [6]: in the first row filters learned in layer 2 are depicted, in second filters of layer 3 are depicted.

variety of both colinear features as well as corners and junctions in the second layer of their architecture (see Fig. 5).

Learning invariant feature hierarchies using convolutional network architecture is presented in [67]. It is shown that a convolutional network produces more diverse and less redundant filters than a non-convolutional network. Similarly, in [6] a convolutional deep belief network is trained on Caltech-101 object recognition dataset. In Fig. 6 we can see that the second layer of the network learns visual parts that are shared amongst object classes. In the same figure we see that the third layer seem to learn filters that are specific to individual object classes.

Recently, deep learning has become popular in the area of saliency prediction too, applied either in an unsupervised way [71] or in a supervised way using recorded ground truth fixations obtained by gaze tracking [29, 30, 31, 32, 72]. [71] use a

simple k-means algorithm in order to learn adaptive low-level filters. In the further processing thresholding and pooling techniques are applied in order to generate more robust mid-level features. Further on, several saliency maps are obtained by using predefined set of hand-crafted filters. Finally, they are fused in order to construct the saliency map. Here, human fixations were not used at any stage of learning. [30] uses the architecture and learned filters of the Krizhevsky network[62]. Those filters are not adapted, i.e. finetuned, using the saliency dataset on which the proposed network was evaluated. However, the weights of the final softmax layer are learned in a supervised manner using human fixations of the dataset on which the approach is validated. [31] and [32] use a multi-layer sparse network to learn low-, mid- and high-level features from natural images and train a linear SVM to learn optimal weights for feature integration, as in [36]. Here, human fixations are used to learn the SVM parameters, and the network parameters are learned using the sparseness criterion. [29] construct multiple layers of biologically inspired features which are (depending on the number of layers) parameterized by various number of architectural parameters. Upon those features a linear SVM is used in the same way as in [36] and [31, 32]. However, here, as opposed to [31, 32], human fixations are used to learn both the SVM parameters and architectural parameters. In the most recent work of [72] human fixations are also used to learn both the SVM parameters and network parameters.

### 2.1.3 Datasets

Here, we will describe the datasets which we will use in order to validate our saliency prediction models, both for images (2D) and video sequences (3D). An extensive overview on eye tracking datasets, both for images and video sequences is given in [73]. For the problem of saliency prediction in 2D images we will focus on the most popular datasets that consist mostly of natural scenes images. For the problem of saliency prediction in video sequences we will focus on video sequences depicting human actions.

#### 2D eye tracking datasets

For the problem of 2D saliency prediction, we will validate our method on three commonly used datasets that contain human eye fixations recorded while the subjects observed images of natural scenes:

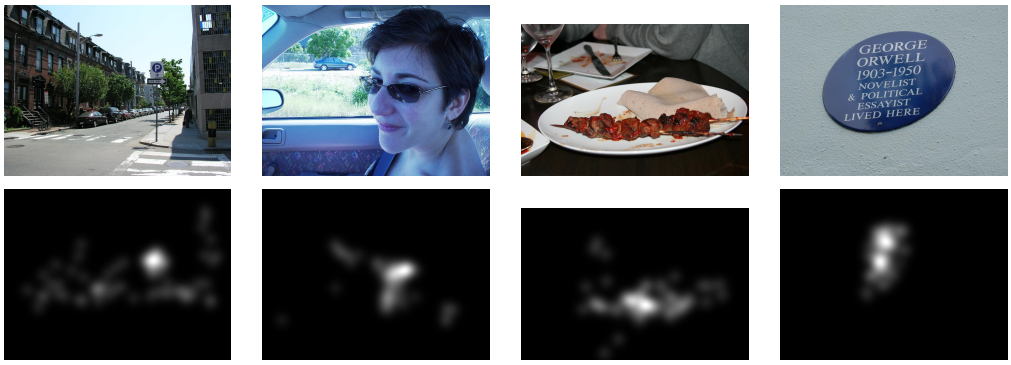



Figure 7: Image stimuli and its corresponding fixation density maps in the MIT dataset

1. the dataset from Judd *et al.*[36] (MIT dataset),
2. the dataset from Bruce and Tsotsos[25] (Toronto dataset),
3. the dataset used in [35, 33, 34] (Kienzle dataset).

In all datasets subjects observed images in a *free-viewing* manner, i.e. without any given task. This suggest using the bottom up saliency detection approach.

The MIT dataset consists of eye tracking data from 15 subjects free viewing 1003 color images of outdoor and indoor scenes. There are 779 landscape and 228 portrait images. Human agreement of this dataset is 90.8%[2]. The examples of image stimuli and its corresponding fixation density maps can be seen in Fig. 7. Fixation density maps are obtained by smoothing the recorded fixation maps using Gaussian kernel with bandwidth set to the visual angle span of 1°.

The Toronto dataset consist of eye tracking data from 11 subjects free viewing 120 color images of outdoor and indoor scenes. A large portion of the images do not contain a particular region of interest (e.g. faces or a distinct object) and therefore it is considered a more difficult dataset. This is probably the reason why human agreement on this dataset is a bit lower, namely 87.8%[2]. This is in agreement with the argument of [39] where it was found that the scanpath consistency in images with faces was higher than in the ones that do not contain faces. The examples of image stimuli and its corresponding fixation density maps can be seen in Fig. 8.

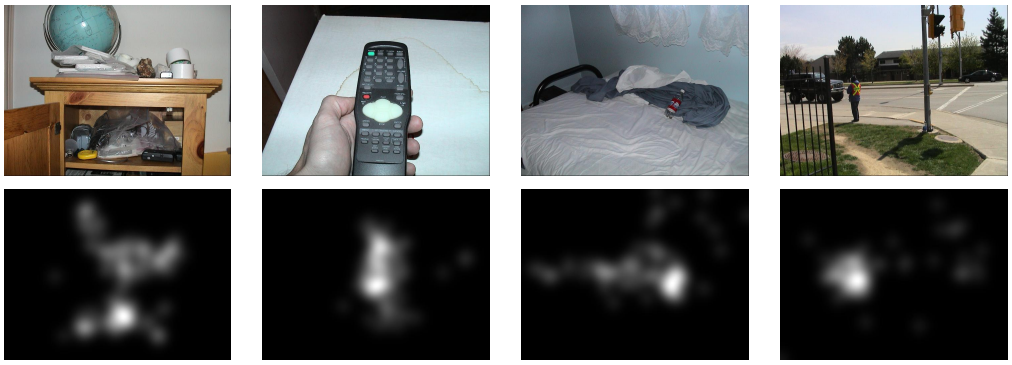


Figure 8: Image stimuli and its corresponding fixation density maps in the Toronto dataset

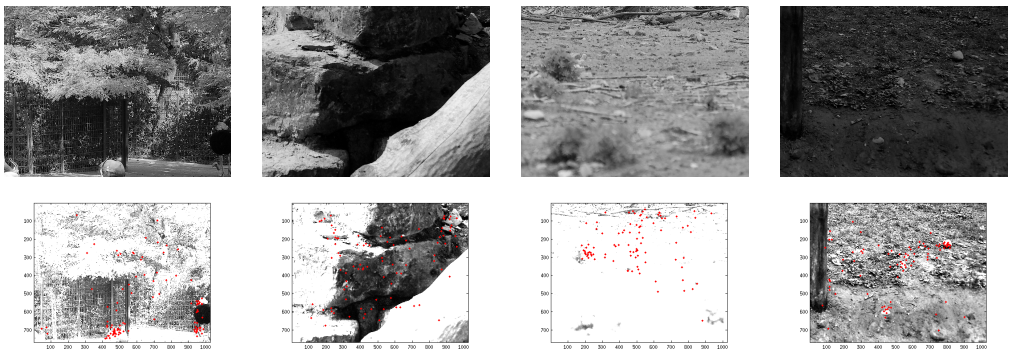


Figure 9: Image stimuli and its corresponding fixations in the Kienzle dataset

The Kienzle dataset consist of eye tracking data from 14 subjects free viewing 200 gray scale images taken in the nature. This dataset is the most challenging since it contains scenes that are very monotonic and without any particular regions of interest. The examples of image stimuli and its corresponding recorded fixations can be seen in Fig. 9.

### 3d (spatiotemporal) eye tracking datasets

The pioneering work on spatiotemporal saliency prediction in action videos is the one of [43]. In this work human eye-tracking annotations are collected for two popular action datasets: Hollywood2 and UCF Sports. The human eye movements were

Figure 10: One example of each action from the UCF sports dataset with the corresponding recorded human eye fixations

collected from 16 subjects that were split in 2 groups: active (12 subjects) and free viewing (4 subjects). The active group had to watch the videos as they were solving action recognition task, while a free viewing group was not given any specific task while watching videos.

There are two important findings in this work. First, it is found that there is a good inter-subject fixation agreement, which makes suitable to use the recorded gaze fixations as ground truth for saliency prediction. Second, it is found that there are no significant differences between the eye movements of active group and free viewing group, i.e. there is no major task influence. This suggest that recorded fixations of all 16 subjects can be used as a ground truth in the same way, as it is in [43, 74]. In our work, we will do the same. The examples of actions and corresponding recorded fixations can be seen in Fig. 10.

### 2.1.4 Conclusions

We have seen how deep learning aims to imitate the human visual pathway by learning features in a hierarchical way - in the same way we aim to build a hierarchy that will be able to use those learned features for saliency prediction task. As topographic ICA has shown to be a good way of modeling the visual complex V1 cells, we will first (in 3.1) investigate how features obtained by using its unsupervised learning criterion at lower layers can be used for saliency prediction.

However, the results presented at the end of 2.1.1 suggest that the focus in training deep architectures should be not so much on the unsupervised criteria, but on the choice of the architecture (its depth, filter size/number, using convolution) and on acquiring more labeled data, since the supervision criteria is what matters. Additionally, the works presented at the end of 2.1.2 suggest that architectures that are optimized for parameters on all layers using supervision from fixations yield better results than the ones that at certain layers use handcrafted features and/or features learned in an unsupervised manner. Also, greater improvements seem to occur when supervision is added in the upper layers.

Therefore, we will further investigate the importance of using the supervision criteria at different levels: first, by backpropagating the supervision in addition to the unsupervised criterion in the lower layers (in 3.2), and second, by using only the supervision criterion on all layers and introducing some architectural changes (in 3.3).

## 2.2 Human action recognition

There is a large body of works in the area of action recognition - for recent surveys we refer the reader to [75, 76, 77]. In what follows, we briefly review some of the major works in the field and then focus on works that are closer related to the contributions that we make, namely, works on feature learning (in 2.2.1), works on joint localization and recognition focusing on latent SVM formulations (in 2.2.2), and finally, works that use gaze in the action recognition framework (in 2.2.3).

In the classical human action recognition pipeline the first step is feature extraction. Usually features based on shape, such as HOG and Scale-Invariant Feature Transform (SIFT), or motion, such as HOF and Histogram of the Oriented edges of the Motion Boundaries (MoBH), are extracted across the areas detected by local

STIP detectors. STIP detectors can be handcrafted (such as 3D Harris detector[45] and periodic detector[44]) or modeled in a generative[78] or discriminative[7] manner. Gaze has been also used as a STIP detector[43, 18, 3]. In such approach, both the feature detector and feature descriptor usually act locally. Detailed analysis on combining various types of STIP detectors and local descriptors is given in [79]. Lately, tracking and extracting trajectory features has shown very good performance in the action recognition[80, 81, 82, 83, 84, 85, 86, 87]. The second step is feature encoding. Until recently a simple BoW approach was most popular: the features extracted around detected areas are quantized and a BoW representation of the whole video is built. Lately, using Fisher Vector (FV) encoding [88] has shown superior results comparing to the BoW approach, in image recognition[89] and action recognition[90, 91, 83, 84, 86, 87]. In the third step an SVM is used as a classifier on top of the video representation.

In such approach local changes can be captured but more complex global spatiotemporal relations and higher level motion patterns are lost by pooling in the feature encoding stage. For that purpose, probabilistic graphical models such as Conditional Random Fields (CRFs)[92, 93, 94, 95], Hidden Markov Models (HMMs)[96, 97, 98, 99], probabilistic Latent Semantic Analysis (pLSA) and Latent Dirichlet Allocation (LDA)[78] can be used. Action grammars[100, 101, 102, 103], models that use graph relations[85, 27] and latent SVMs[104, 74, 105] are also used to model the higher levels of action recognition frameworks.

In 2.2.1 we will present works that address the problem of finding good features for action recognition. In 2.2.2 we will present methodologies for action recognition where action classification works on detection of relevant parts of the action videos and higher modeling of their relations. In 2.2.3 we will present the work in which gaze information has been used in action recognition. In 2.2.4 we will present human action dataset we will use in our experiments.

### 2.2.1 Feature learning for action recognition

Much effort has been put in enriching local descriptors and detectors with for example, hierarchical structures [106, 7, 107], local contexts [79, 108], or extending them to 3D [109, 110, 111]. As mentioned previously, an approach that is focused on learning

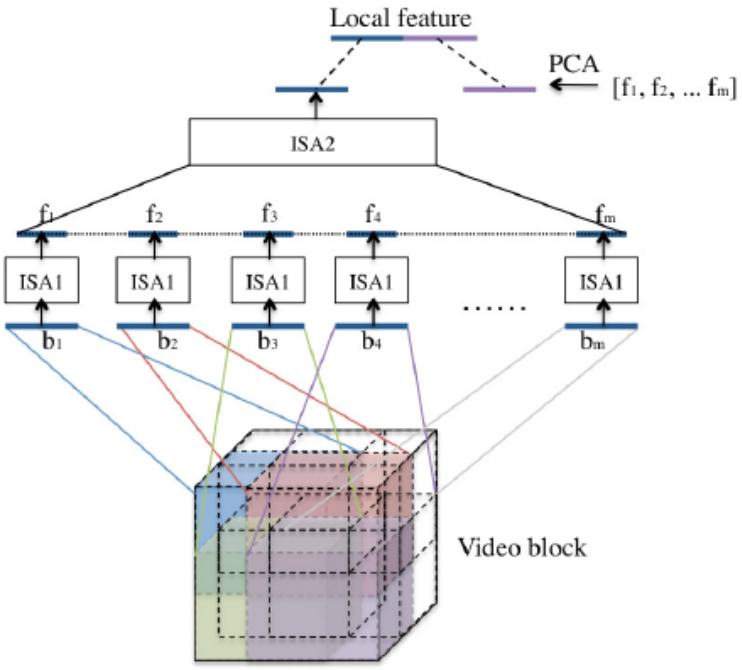



Figure 11: Stacked convolutional ISA for videos[7]

feature representations, namely deep learning, has become very popular in the last few years, and has shown very good performance in various computer vision tasks [62, 6, 63], including action recognition [7, 112, 113, 64, 114].

In [7, 114] features for action recognition are learned from the video data using unsupervised learning algorithms: [7] uses Independent Subspace Analysis and [114] uses Slow Feature Analysis. (ISA algorithm was mentioned in 2.1.1.) In both of those works, features are learned in two layers using common deep learning techniques: convolution and stacking. In Fig. 11 we show the learning architecture used in [7] and in Fig. 12 we show the visualizations of spatiotemporal filters learned in the first and second layer. Note that the authors visualize the *optimal stimuli* for the second layer features, as visualizing and analyzing higher layer units is usually difficult. Such features consistently outperform handcrafted ones in a BoW framework. Interestingly, [7] also show very good generalization ability of their learned features, i.e. even if features are learned using one type of data, they perform well on unrelated data as well.

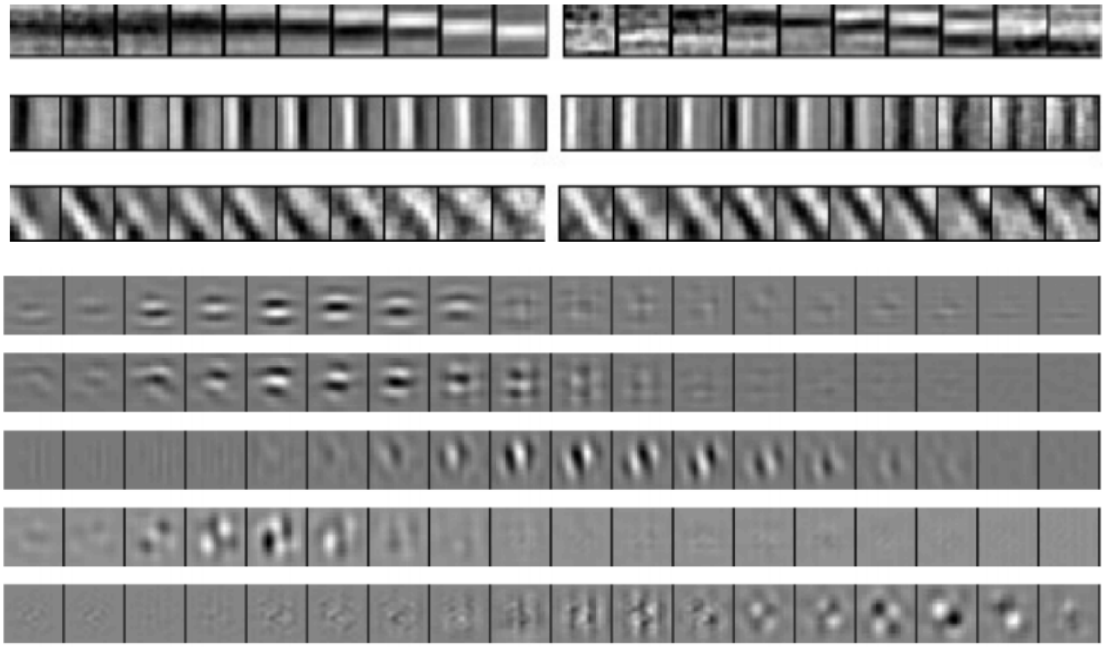


Figure 12: In the first three rows each row consists of two sets of 3D first layer filters. Rows 4-8 consist of of five 3D optimal stimuli in the second layer.

Another work that uses unsupervised deep learning approach for action recognition is [115]. In order to learn spatiotemporal features they use convolutional gated RBMs. As a results, useful motion-sensitive features, as well as segmentation and edge-detection operators are extracted.

Supervised deep learning algorithms, such as 3D CNNs, have also been successfully applied for the problem of action recognition [64, 113, 112, 116, 117]. The work of [64] is the first one in which a 3D CNN is applied for the problem of action recognition. What is particularly interesting in that work is that it shows that 3D CNN outperforms classical methods in real-world environments, while giving comparable results when applied in the controlled environments. However, to make the training of the full network easier, in that work handcrafted features are used at the bottom most layers. The work of [113] uses 3D CNN in the same way as [64], the only major difference being that they do not use any handcrafted features: their whole architecture is trained in a supervised way. In addition to that, on top of the extracted 3D

CNN discriminative features they use another discriminative type of neural network which is typically used for sequence classification, namely LSTM-RNN (Long Short Term Memory Recurrent Neural Network)[118]. There are two other works from the same authors in which similar approach is applied: [119, 120]. In [119] the features that are fed into a LSTM-RNN are extracted by a sparse 2D autoencoder. In [120] 2D convolutions are extended to 3D. This improved the results.

The works regarding supervised 3D CNNs described so far are applied in a holistic manner, i.e. CNN features are learned using whole video sequences as an input by gradually increasing the level of the features from local to global. Therefore, to reduce the input dimensionality and to remove the background noise those networks require the inputs to be segmented video sequences. Also, they are mostly evaluated only on the KTH dataset.

Recently, there has been much advances in applying deep CNNs for action recognition [117, 112, 116, 87, 121]. Newly developed architectures do not require the inputs to be segmented and they are evaluated on larger datasets, such as UCF101[122], HMDB-51[123] and Sport1M[112]. In [117] a two stream architecture that is supposed to capture the complementary information on appearance from still frames and motion between frames is trained. This architecture achieves competitive results on UCF-101 and HMDB-51 datasets and exceeds by a large margin previous attempts to use deep networks for video classification. Also, in contrast to previous works, their network is trained on full frames. However, their architecture uses only 2D convolution and 2D pooling operations. In contrast to this, [116] learn spatiotemporal CNN features using 3D convolutional filters and pooling operations upon sequences on full video frames. They show that such features in combination with a linear SVM outperform state-of-the-art results on few benchmark datasets. [112] explore various approaches for fusing information over temporal dimension through the network, i.e. they use both 2D and 3D convolutional architectures and show that the architecture that uses 3D convolutions performs best in terms of action classification accuracy. They also show that their learned features are able to generalize well by performing a transfer learning experiment in which only higher level features are fine-tuned when applied on different dataset. Those results are in line with the claim of [7] about the generalization ability of their features. It is also becoming common to combine deep learning features with some classical action recognition approaches,

such as trajectorie-based[87], or pose-based[121] methods.

As we mention, some not-so-recent action recognition models [113, 119, 120] use a neural network that is developed particularly for sequence classification, that is LSTM-RNN. However, their methods are evaluated only on KTH dataset. Nowadays, LSTM-RNNs are gaining more popularity even in large scale action recognition[124, 125, 126]. [124] try to learn the dynamic saliency of the actions in the context of LSTM RNNs by detecting and integrating the salient spatio-temporal sequences. They show that the results they obtain outperform the standard LSTM RNN model. Conceptually similar to this work is the work of [126]. Their model learns to focus selectively on parts of the video frames and classifies videos using those parts as an input to the LSTM RNN. In contrast to this, [124] is learning the salient parts of the videos by changing the intrinsic properties of LSTM RNNs neurons. This means that [124] learns spatiotemporal saliency while [126] learns only spatial saliency. [125] address the problem of learning a global description of the videos temporal evolution. For this purpose they present two video-classification methods capable of aggregating frame-level CNN outputs into video-level predictions. One of them is based on feature pooling and another one on LSTM RNN. Using those methods they achieve state-of-the-art performance on Sports1M and UCF101 datasets.

We will also mention a couple of works which do not use any of the standard deep learning architecture but adopt the two level discrimination approach. In [127, 128], inspired by [129], the discriminative mid-level features are learned using an exemplar SVM on manually designed low-level features. Mid-level features are used to build a global video representation which is then fed into a final classifier, also an SVM. In [130] discriminative mid-level features are learned using random forests classifier on optical flow and histograms of 3D gradients. The mid-level features are then fed into a second random forests classifier. However, in both of those works additional information about discriminative parts of the video is necessary. In [127, 128] discriminative parts of the training videos are manually selected, and in [130] a human detector is used to determine the discriminative area across which the mid-level features are extracted and fed into a subsequent random forest classifier.

### 2.2.2 Latent variables for action recognition

In this section we will focus on the approaches that use latent variables as a higher level modeling in the action recognition framework.This approach is inspired by the work of Felzenschwalb *et al.*[1] where latent SVM is used in the task of part-based object recognition. In the same work it is shown how semi-convex optimization problem of the latent SVM becomes convex when latent information is specified for positive examples.

Some works that build up on this work in the domain of, for example, image classification [131] or object detection [132] also show good results. In [131] image classification problem is cast in a latent SVM framework that uses image saliency map as a latent variable. An image is represented by concatenation of local bag-of-features weighted by the image saliency maps. It is shown that using saliency maps to weight the corresponding visual features improves the discriminative power of the image representation.

In the action recognition there are works which employ latent variable modeling: [133, 99] use latent variables to model the temporal structure of activities and [104, 105] try to jointly localize and recognize the actions in a latent SVM framework, i.e. they use latent variables to model the spatial structure of activities. In [105] a video sequence is, in addition to local features that appear in the entire sequence, represented by a latent region of interest within a sequence. Previous work of the same authors [104] tries to jointly capture the relationship between the action label and the location of a human performing the action by introducing a latent variable that discriminatively selects relevant parts of the humans bounding box. In [134] a joint learning process for localization and classification of both static images and time series data is presented. In contrast to [104, 105] which use latent SVM, in [134] this is achieved by adding constraints in the standard SVM optimization problem. The constraints state that each positive example must contain at least one subwindow classified as positive and that all subwindows in each negative example must be classified as negative.

Finally, a recent work that is the closest to the approach we will adopt is [74]. This work builds up on [104, 105] where structured output latent SVM for action recognition is presented. Additionally, in [74] recorded human gaze is incorporated

in the SVM training procedure. Therefore, this work will be described in more detail in 2.2.3.

### 2.2.3 Human action recognition using gaze information

There are works that use the information from recorded human eye movements in order to obtain some clues to either localize the relevant parts in images/videos[16, 17, 19, 43, 18, 14], or to infer the relevance of the observed images/videos[11, 12, 13] for which the eye movements were recorded.

To verify if the gaze information could be of any use in such tasks, it is necessary to perform an analysis of gaze patterns. As we mention in 2.1.3 the work of [43] recorded gaze movement for a couple of action datasets and found that there is a good inter-subject fixation agreement, which made suitable to use the fixation data as a ground truth STIP detection. [10] analyze gaze patterns in the context of image classification. Main findings of this work are: (1) for some images there are similar viewing patterns, but for some not, (2) the same subject uses different patterns on different images, and (3) there are differences in viewing patterns across categories. For example, the goal of [11, 12, 13] is image/video classification and (3) implies that gaze patterns can be used for this purpose. However, [11], [12] and [13] use different type of data for which the findings presented in [43] and [10] may not completely hold.

Promising results on using gaze movements in video annotation are reported in [13]. Here, gaze-based features for each video are extracted and used to train a classifier that can classify videos as relevant or non relevant to a certain topic. Such classifier trained only using gaze movements can be considered as predictor of user interest for a certain video in the context of a query topic.

Eyetracking methodology has been used in the field of image/video retrieval too. [11, 12] are trying to infer the relevance of the images based on implicit feedback from users attention measured using an eye tracking device. Usually, in the relevance feedback process user is asked to refine the searching by providing the feedback explicitly, for example by selecting Areas-of-Interest (AOIs) from the query image, or to tick positive and negative samples from retrieves. This process is, however, laborious for the user. Therefore, eye movements implicit feedback is proposed as a rich

and natural source to replace the time-consuming and expensive explicit feedback. In those works promising results are obtained using a classifier that is trained upon a set of simple eye movement features. Furthermore, in [11] the classifier is able to detect the relevance even when not using any training data from particular user in question, which suggests that gaze patterns of different users are similar enough for this task. This is in line with the statement of [43] about good inter-subject fixation agreement on their video data.

There is another example of using gaze information for a segmentation problem: in [14] gaze information is employed for the problem of interactive medical image segmentation. Here, gaze information is used for placing object and background seeds instead of traditional mouse moving and clicking. It is shown that the resulting interaction is either comparable to or an improvement over existing input methods and that it is indeed possible to devise novel interaction techniques that use gaze as a form of input for interactive image segmentation, even given the unfamiliarity of the subjects with the eyegaze interface compared to the mouse.

For videos containing human actions it is useful to determine interest points where features should be extracted and further used to train the classifier. Salient point techniques, i.e. interest point operators, are often used to represent the information about area relevance in terms of saliency maps. Using this information, more discriminative features can be computed[135, 136]. However, as interest points detector usually focus on local instead of global features, the detection of spatiotemporal interest points on human bodies in complex scenes or on cluttered backgrounds may fail.

Improvement of saliency maps can be achieved by employing human vision system. It has been shown how both 2D and 3D (spatiotemporal) salient point detectors can be learned from human eye fixations[36, 33, 34, 2, 37, 35, 3, 43, 42]. In particular, this has been shown for videos that contain humans performing actions[43].

Most of the works use gaze information in the action recognition framework as a STIP detector[43, 18, 3]. This can be done in two ways: first by using ground truth recorded fixations as a STIP detector in test videos [18], and second by using ground truth fixations to train a detector on a training set and use the predicted fixations as a STIP in test videos [43]. [18] showed that results for action recognition obtained using the former approach outperformed the state-of-the-art results

obtained using commonly used STIPs in combination with HOG-HOF features[79], dense trajectories[80] or stacked ISA features[7]. However, this method requires gaze information in test videos and this is not always available.

[43] and [3] learn to predict the human fixations on test videos using recorded fixations only on training videos. As mentioned in 2.1, [3] is the first work that introduces the concept of using human ground truth in spatiotemporal saliency learning and uses the learned saliency in the action recognition framework. They show that action recognition results when using the proposed neural network as spatiotemporal point detector outperform the state-of-the-art methods which use handcrafted spatiotemporal detectors. Learning saliency prediction in this work is described in 2.1. In the same section we describe the saliency learning presented in [43]. In [43] it is first shown that action recognition results obtained when ground truth fixations as a STIP outperform the results obtained when using using common STIPs in a classical STIP+BoW+SVM pipeline. This is consistent with the results of [18], where the same action recognition datasets is used (although, [18] recorded their own eye tracking data). Second and more importantly, it is shown that action recognition results obtained when using fixations predicted by the learned saliency predictor as a STIP outperform the results obtained when using common STIPs in a classical STIP+BoW+SVM pipeline. However, they do not outperform the ones obtained when using ground truth recorded fixations as a STIP.

Finally, a work that is closest to ours is [74]. Here, recorded human gaze is incorporated in a structured output latent SVM framework. The gaze was used only in the training phase in a form of a loss in the structured prediction in order to infer the latent variable that determines the bounding boxes of the action. This kind of formulation leads to a difficult optimization problem since in the inference there is a large search space over all possible spatio-temporal paths. Such formulation also allows a top-down saliency prediction within the predicted bounding boxes. Saliency prediction results obtained using this kind of top-down saliency prediction model have shown to be comparable to the ones obtained using the bottom-up approach of [43].

There is another approach in using gaze for 2D action recognition presented in [19]. That is using gaze features to train an action classifier, similar as the works we mention at the beginning of this section[11, 12, 13] use gaze features for various classification tasks. In [19] action classifiers are trained using gaze features and CNN features. It

turns out that gaze features do not outperform CNN features, but combining gaze features with CNN features does yields improvement over the approach that uses only CNN features. In [16, 17] the same authors propose a similar gaze-enabled object detection scheme for 2D object detection. Their proposed scheme uses gaze features to train classifiers that should distinguish between true positive and false positive detections generated by baseline detectors. They show that their proposed detection scheme outperforms the object detection schemes which do not use gaze information. However, all those works ([19, 16, 17]) require gaze in the testing phase.

### 2.2.4 Datasets

We will validate our proposed action recognition methods on two sports action datasets:

1. UCF sports dataset[137],
2. Oympic sports dataset[133].

On the UCF sports dataset we will validate both saliency prediction and action recognition methods. For the Olympic sports dataset there are no recorded gaze fixations available; therefore we will use it to validate only action recognition methods.

UCF sports is a high resolution dataset collected mostly from broadcast television channels. It contains 150 videos depicting 10 sports actions classes: diving, golf swinging, kicking, lifting, horseback riding, running, skateboarding, swinging on the bench, swinging in the air and walking. The actions are recorded in different scenes and from different viewpoints. Some of the sequences contain more than one human, although only one of them is performing the action. The examples of actions and corresponding recorded fixations can be seen in Fig. 10.

The Olympic sports dataset contains videos of athletes practicing different sports. There are 783 videos of 16 sports action classes: high jump, long jump, triple jump, pole vault, basketball lay-up, bowling, tennis serve, platform diving, discus throw, hammer throw, javelin throw, shot put, springboard diving, snatch (weightlifting), clean and jerk (weightlifting), gymnastic vault. Videos are collected from YouTube and class labels are obtained with the help of Amazon Mechanical Turk. The examples of actions can be seen in Fig. 13.

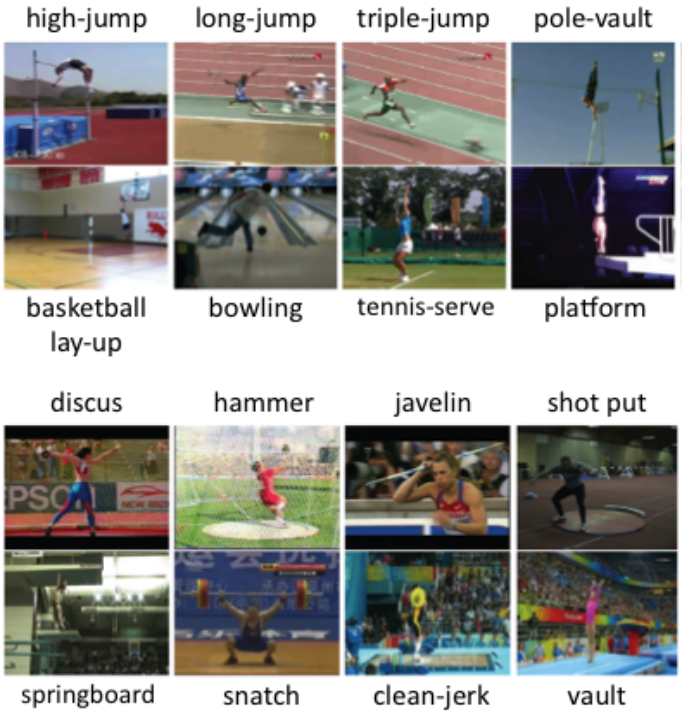



Figure 13: Examples of actions in Olympic sports dataset

### 2.2.5 Conclusions

The works presented in 2.2.1 suggest that replacing commonly used manually designed features with features that are *learned in a supervised way* is a good way for future research in the action recognition and many other areas. In the same section we have seen that commonly used BoW approach and other works with a similar pipeline lack the knowledge about discriminative parts of the videos and in some of them, for example [127, 128] and [130], additional annotations are necessary.

The works presented in 2.2.2 aim to search for the discriminative areas during the training of an (action) classifier. This leads to a challenging optimization problems. However, having gaze information available in the training phase, it might be possible to alleviate/improve the learning of a classifier by using discriminative areas inferred by gaze fixations.

In the works presented in 2.2.3 gaze information is used as an additional clue about the action location. However, in all the works except in [74] gaze information

is used only in the process of sampling salient points which are then fed into an SVM classifier, i.e. except in [74], gaze information is not incorporated in the higher levels which include training a classifier. There are no works regarding action recognition in which gaze information is incorporated in the feature extraction at lower layers.

We aim to use gaze information only in training, as in [3, 43, 74] and we aim to incorporate it in both lower and higher levels of our framework. First, as the deep learning approach shows good results, we would like to employ it and the human gaze in order to learn discriminative features in the lower layers. Second, since introducing latent variables for action localization has shown promising results, we aim to use a similar framework and incorporate the loss from saliency prediction in it. Rather than using a purely top-down saliency prediction approach as in [74] we will build a bottom-up saliency detector and incorporate its loss in the SVM training procedure.

# 3 Visual saliency learning

In this chapter we will present our saliency prediction models that learn to predict where humans look using recored gaze information as a ground truth when training a classifier that learns to classify fixations and non fixations. As opposed to previous works that also use gaze information as a ground truth only when training a classifier, we will use supervision criterion derived from gaze fixations in order to learn features at different levels. No handcrafted features will be used. Learning features will be first performed in separate phases, but we will investigate how joint learning of lower and higher layers affects the saliency prediction results. Finally, we will present an architecture in which all layers are optimized jointly.

In the first phase we will learn in an unsupervised way low level features that will be fed as an in input to an SVM classifier. Instead of using combination of handcrafted low level features we propose using features obtained by topographic Independent Component Analysis. Topographic ICA is a generalized version of Independent Component Analysis where higher order dependencies define a topographic order such that near-by cells tend to be active at same time[46]. However, when learning a bottom-up saliency we do not consider that some shapes are not just blobs or lines, but they carry some semantic meaning for us, like for example faces. An extensive discussion on that is given in [39]. It is shown that in the saliency prediction task face detection in combination with low level features yields significant improvements over low level features only.

In the second phase we will jointly learn the higher and mid layers using the supervision criterion derived from the SVM formulation and gaze fixations. We will do so by learning the pooling operation in tICA, i.e. the weights of the second layer in the topographic ICA architecture which are originally set to be fixed. In order to learn the pooling we will use linear SVM supervision criterion and see how this can help us building a better overall architecture.

Third, we will use a fully supervised 2D convolutional neural network architecture where all layers are optimized jointly using only supervision criterion. We will see how the results of such architecture compare to the one where tICA criterion is applied at the lowest layers in a separate learning phases while supervision being incorporated only in the highest level, and to the one where supervision incorporated in the highest

mid layers using a joint optimization procedure. Furthermore, we will extend this 2D convolutional architecture to 3D, apply it in the spatiotemporal domain, i.e. videos that contain human actions, and see how 3D architecture compares to 2D architecture when used for the problem of spatiotemporal saliency prediction.

The contributions of this chapter are the following:

1. We have proposed a method that learns bottom-up visual saliency by learning features from the given images instead of using manually selected ones. By doing so, we did not make any assumptions about occurring image structures, rather we infer them from training images. The saliency prediction results have shown that the performance of our model was comparable to sophisticated approaches where higher level features such as face detectors were used.

2. We have proposed a novel method for learning the pooling of topographic ICA using linear SVM supervision criterion and shown that it led to improvements in the saliency prediction results over using topographic ICA features with fixed pooling and linear SVM on top of them. Furthermore, we have shown that a convolutional architecture that uses supervision criterion at all layers outperformed other two proposed architectures which do not use the supervision criterion at all layers.

3. We have shown that extending 2D convolutions to 3D improves the saliency prediction in the spatiotemporal domain.

The rest of the chapter is organized as follows. In 3.1 we will describe in more details the model of topographic ICA. In 3.2 we will describe how we learn the pooling of topographic ICA using linear SVM criterion. In 3.3 we will describe how we apply 2D and 3D convolutional neural networks for the problem of spatial and spatiotemporal saliency prediction. In 3.4 we will show the results of the experiments we performed on three different 2D dataset and one 3D (action videos) datasets, comparing the results obtained by using different architectures and optimization criteria. In 3.5 we will present the conclusions.

## 3.1 Topographic Independent Component Analysis

Topographic Independent Component Analysis was originally proposed in [46] as a generalization of linear Independent Component Analysis and a biologically plausible way of modeling complex cells of human visual V1 area. Here, we give a brief overview of the topographic ICA model and stress how it differs from linear ICA. A detailed analysis is given in [8, 47, 46].

tICA is an unsupervised learning algorithm that learns features from unlabeled image patches. It can be described as a two-layered network with squared nonlinearity in the first, and square-root nonlinearity in the second layer of the network (see Fig. 14).

Assuming that we have observed a set of image patches $\mathbf{z}^t \in R^{n_0}, t = 1, ..., T$ after vectorization and whitening, tICA learns $\{\mathbf{v}_j\}_{j=1}^{n_1}$ by solving the following optimization problem:

$$\min_{\{\mathbf{v}_j|\ j=1,...,n_1\}} \sum_{t=1}^{T} \sum_{i=1}^{n_2} \sqrt{\sum_{j=1}^{n_1} \pi(i,j)(\mathbf{v}_j^T \mathbf{z}^t)^2} \tag{2}$$

where $n_0$ is the patch dimensionality after vectorization and whitening, $n_1$ is number of the neurons in the first layer and $n_2$ is the output dimensionality (i.e. number of neurons in the second layer after the pooling). In addition, the vectors $\mathbf{v}_j \in R^{n_0}, j = 1, ..., n_1$ are constrained to be orthogonal. Minimization of this function is done by batch gradient descent. The topography (pooling matrix) given by $\pi(i,j) \in R^{n_1} \times R^{n_2}$ is considered fixed and only the linear feature weights $\mathbf{v}_j$ need to be estimated, i.e. a feature representation is learned only through the weights of the first layer, that is $\mathbf{v}_j$. The matrix $\pi(i,j)$ expresses the proximity of the features with indices $i$ and $j$ in a predefined underlying topography. The features are arranged in a 2D lattice and square neighbourhoods are defined. For example, $\pi(i,j)$ is 1 if the feature $j$ is in a 3x3 square neighbourhood of feature $i$; otherwise $\pi(i,j)$ is zero. (see Fig. 15; $s_i = (\mathbf{v}^T \mathbf{z})^2$; $\mathbf{s} \in R^{n_1}$). This type of arrangement restricts features in the same clusters to have adaptive weights $\mathbf{v}_j$.

Once the vectors $\{\mathbf{v}_j\}$ are estimated, given a whitened image patch $\mathbf{z}$, we can extract a feature $\mathbf{x} = [x_1,\ ...,\ x_i,\ ...,\ x_{n_2}] \in R^{n_2}$ (see Fig. 16). The $i$-th element of $\mathbf{x}$

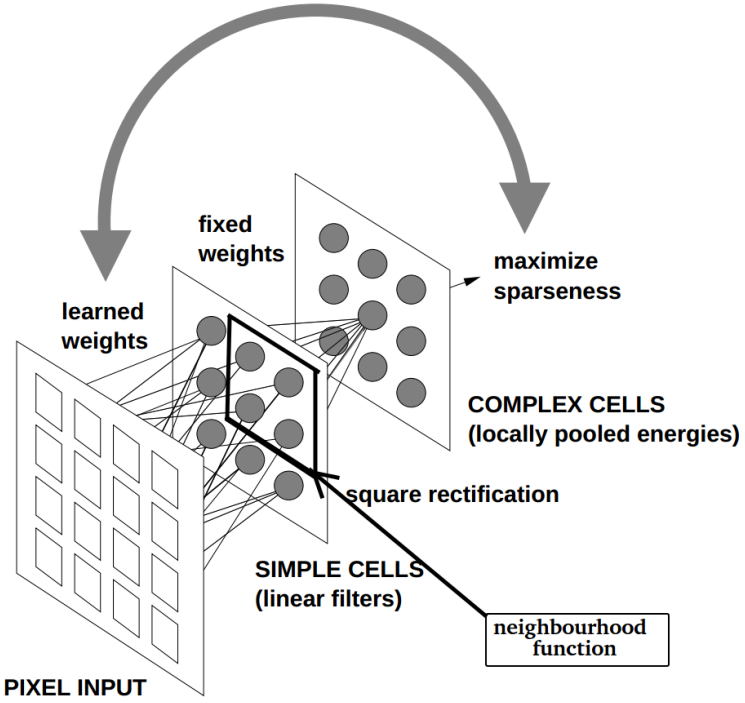



Figure 14: tICA model[8]

is given by:

$$x_i = \sqrt{\sum_{j=1}^{n_1} \pi(i,j)(\mathbf{v}_j^T \mathbf{z})^2}. \tag{3}$$

In Fig. 17 we show the basis vectors we obtained for tICA in order to illustrate the emergence of topographic organization. As we can see, applying ICA leads to emergence of oriented filters that are localized both in space and in frequency, thus resembling V1 simple-cell receptive fields[46]. In contrast to such Gabor-like linear features, in the case of tICA the location and orientation change smoothly in the topographic grid. In a biological interpretation, this kind of topological arrangement is assumed to be useful for minimizing the wiring length i.e. if we assume that two cells need to communicate with each other if (and only if) their outputs are statistically dependent, then tICA provides optimal wiring[8].

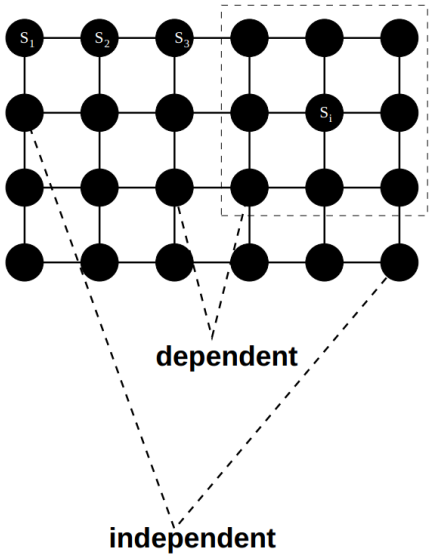



Figure 15: tICA feature detectors are arranged on a 2-D grid and function $\pi$ then defines the neighborhood of these feature detectors as a set of cells that are inside a certain radius[8].

## 3.2 Supervised pooling

Typically, learning saliency is posed as a binary classification problem over some handcrafted features[33, 34, 35, 36, 37, 2]. In this framework, let $\left\{(\mathbf{x}^t, y^t)\right\}_{t=1}^{T}$ denote a set of training vectors $\mathbf{x}^t \in R^{n_2}$ representing the extracted features of image patches (see Eq. (3)) and their corresponding labels $y^t \in \{-1, 1\}$. $y^t = 1$ if the position where $\mathbf{x}^t$ is extracted is a fixation point and $y^t = -1$ otherwise (fixation point are points at which users that were presented with the image in question fixated their gaze; for details see section 3.4 or [36, 25, 35]).

Standard SVM learning problem is posed as the following minimization problem[138]:

$$\min_{\mathbf{w},b} f(\{\mathbf{x}^t\}_{t=1}^{T}; \mathbf{w}, b), \tag{4}$$

where

$$f(\{\mathbf{x}^t\}_{t=1}^{T}; \mathbf{w}, b) = \frac{1}{2}(\mathbf{w}^T\mathbf{w} + b^2) + C\sum_{t=1}^{T}(\max(0, 1 - y^t(\mathbf{w}^T\mathbf{x}^t + b)))^2, \tag{5}$$

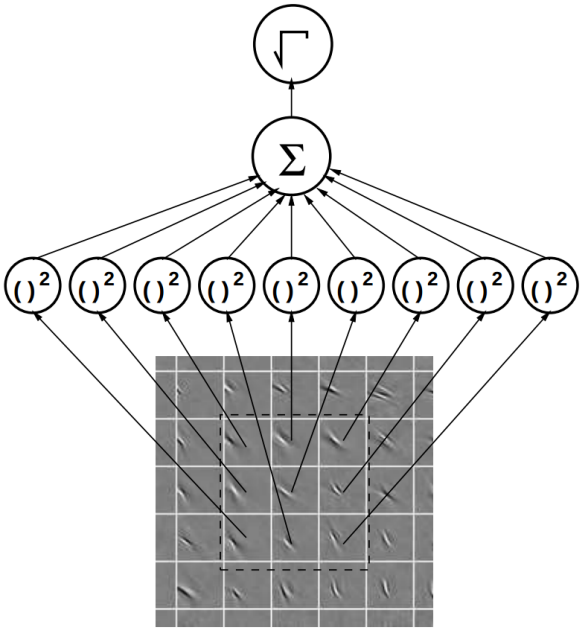
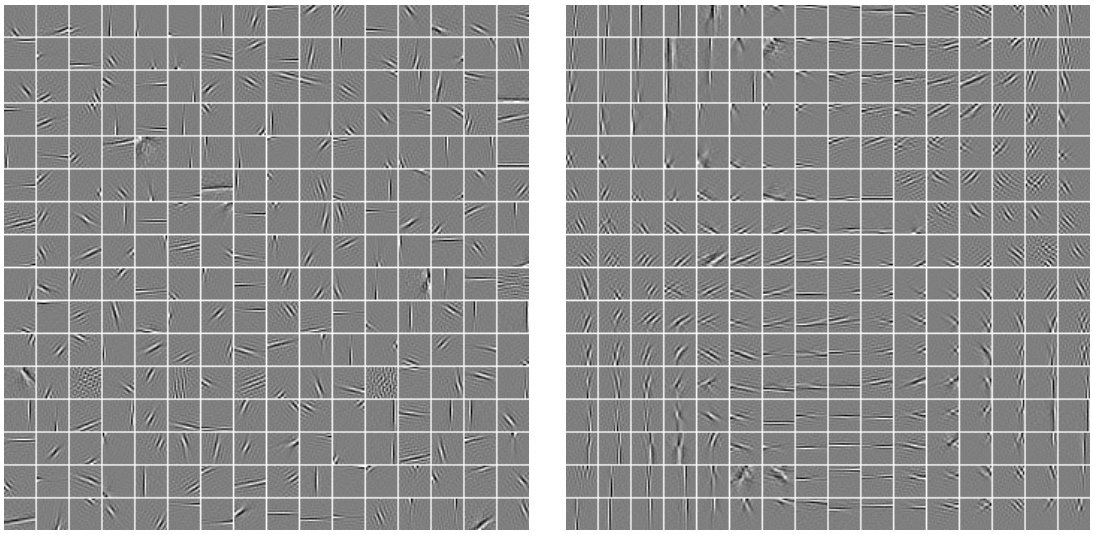



Figure 16: Illustration of feature extraction[8]



(a) (b)

Figure 17: Basis vectors obtained on 21x21 patch size for: (a) ICA, (b) tICA.

where weights $\mathbf{w}$ and bias $b$ are the SVM parameters. In the standard formulation only those are learned. Instead, we solve the following minimization problem:

$$\min_{\mathbf{w},b,\pi} f(\{\mathbf{x}^t\}_{t=1}^T; \mathbf{w}, b, \pi), \tag{6}$$

where

$$f(\{\mathbf{x}^t\}_{t=1}^T; \mathbf{w}, b, \pi) = \frac{1}{2}(\mathbf{w}^T\mathbf{w} + b^2) + C\sum_{t=1}^{T}(\max(0, 1 - y^t(\mathbf{w}^T\mathbf{x}^t + b)))^2, \tag{7}$$

where $\pi$ is the pooling matrix and $\mathbf{x}$ is a function of $\pi$ (see Eq. (3)). We solve this by following a coordinate descent iterative optimization procedure which iterates between steps S2 and S3 (see Fig. 18). Each of those two steps is a convex subproblem solved with respect to a subset of the unknown variables. With our joint optimization procedure we are going to make the weights $\pi(i, j)$ adaptive and dependent on the nature of the supervision criterion, i.e. we will change the non-zero elements of the pooling matrix $\pi(i, j)$, while keeping the size of the neighbourhood fixed, i.e. the zero elements stay zeros.

Similar optimization schemes that are trying to optimize both for the supervised and unsupervised criteria are employed in, for example, [139, 140]. In [139] SVM classification constraints are incorporated in the non-negative matrix formulation. For such formulation an iterative optimization procedure is proposed. In each iteration the procedure optimizes with respect to subsets of the unknown variables, that is the bases, the projection coefficients and the separating SVM hyperplane parameters. A framework that simultaneously learns the data discriminator and data generator parameters is presented in [140]. The generator captures the data distribution, and the discriminator is trained to maximize the probability of assigning the correct label to both training examples and generator samples. Both generator and discriminator parameters are updated in an iterative optimization procedure. The updates are based on the gradients of the generators and discriminators loss functions.

Our full learning algorithm, which consist of three steps, is given below:

S1. Pretrain the lowest layer of the architecture according to the tICA criterion,

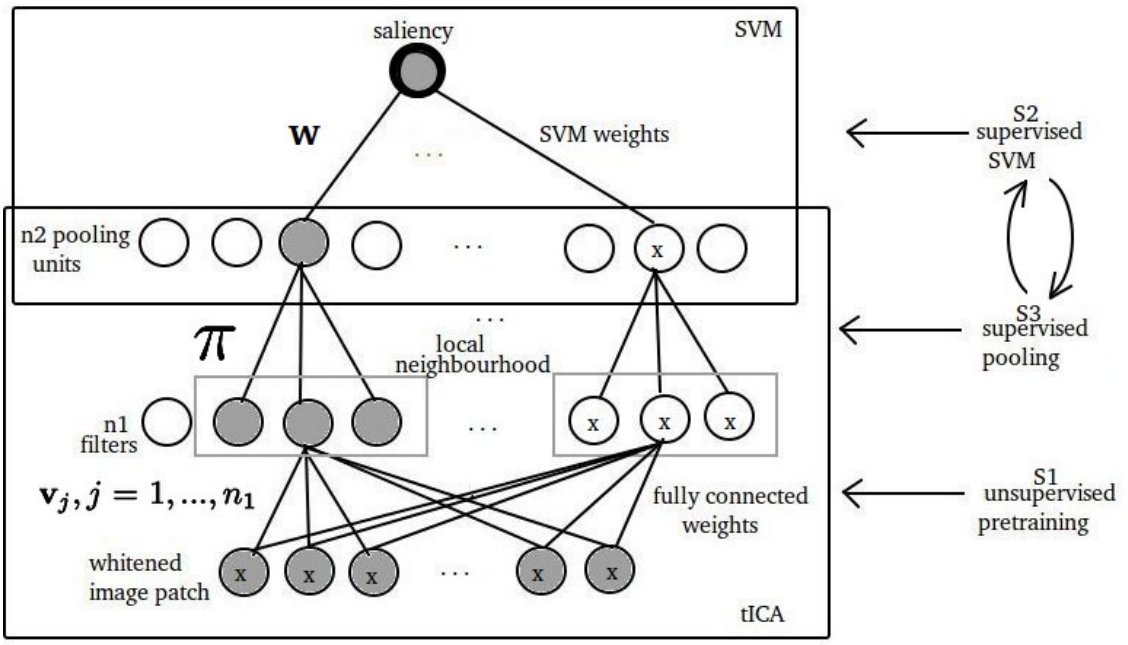



Figure 18: Illustration of our proposed architecture. Due to clarity, weights are shown only for the marked neurons.

i.e. in an unsupervised manner (see Eq. (2)).

S2. Minimize with respect to SVM parameters $\mathbf{w}, b$ keeping $\pi$ fixed. That is:

$$\min_{\mathbf{w},b} f(\{\mathbf{x}^t\}_{t=1}^T; \mathbf{w}, b, \pi). \tag{8}$$

This is a standard SVM problem that we solve using LIBLINEAR library[141].

S3. Minimize with respect to the pooling matrix $\pi$ keeping $\mathbf{w}, b$ fixed. That is:

$$\min_{\pi} f(\{\mathbf{x}^t\}_{t=1}^T; \mathbf{w}, b, \pi). \tag{9}$$

Following the idea of training an SVM in the primal form [142], this optimization is solved using batch gradient descent. In our optimization procedure, one gradient step for updating $\pi$ consists of four substeps:

1. Take a step following the gradient of (9) with respect to $\pi$, that is:

$$\pi^{k+1} \leftarrow \pi^k - \mu \frac{\partial f(\{\mathbf{x}^t\}_{t=1}^T; \mathbf{w}, b, \pi)}{\partial \pi} \tag{10}$$

where $\mu$ is the learning rate. We note that the function $f(\{\mathbf{x}^t\}_{t=1}^T; \mathbf{w}, b, \pi)$ is not differentiable. Therefore, as $\frac{\partial f}{\partial \pi}$ we use its subgradient:

$$\begin{cases} 0, & \text{if } y^t(\mathbf{w}^T\mathbf{x}^t + b) \geq 1, \\ 2Cy^t\mathbf{x}^T\mathbf{w}(1 - y^t(\mathbf{x}^T\mathbf{w} + b))\left\{\left[(\mathbf{V}^T\mathbf{z}) \circ \left(\frac{\pi}{\sqrt{\pi(\mathbf{V}^T\mathbf{z})^2}}\right)\right]\mathbf{z}^T\right\}, & \text{if } y^t(\mathbf{w}^T\mathbf{x}^t + b) < 1, \end{cases} \tag{11}$$

where $\circ$ denotes elementwise multiplication.

2. Constrain $\pi$ to have non-negative elements, that is:

$\pi_j^{k+1} \leftarrow \pi_j^{k+1} + \min_l(\pi_j^{k+1}(l)), j = 1, ..., n_2,$

where $l$ is an index to the elements of the vector $\pi_j^{k+1}$.

3. Preserve the neigbourhood size, i.e. set the elements outside of the neighbourhood to zero, that is:

$\pi^{k+1} \leftarrow \pi^{k+1} \circ \pi_0,$

where $\pi_0$ is the fixed pooling matrix as proposed in [8] (see Fig.15) and $\circ$ denotes elementwise multiplication.

4. Normalize $\pi$:

$\pi_j^{k+1} \leftarrow \frac{\pi_j^{k+1}}{\|\pi_j^{k+1}\|}, j = 1, ..., n_2.$

Typically, the algorithm converges after few iterations between steps S2 and S3 resulting in an energy decrease of 3-5%, depending on the dataset.The number of iterations in substep S3 was experimentally set to 100 for all datasets. The size of the neighbourhood is proposed to be set to 3x3, however, in our experiments we have found that 5x5 gives better results and also better improvement after the joint optimization (since the capacity for learning the weights is larger in the larger neighborhood; however, neighborhoods larger than 5x5 didn't perform well, neither

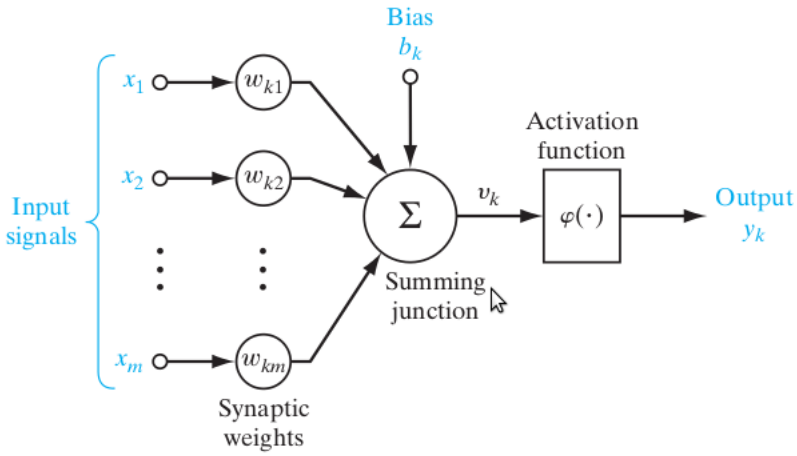



Figure 19: A model of a single neuron $k$ [4]

in the case of fixed nor in the case of learned pooling). Other parameters were chosen either as proposed in the literature [8] ($n_1$ and $n_2$ in tICA network) either by cross validation ($C$ value of SVM). Also, we have found that using a single image patch size (41x41) was sufficient, while in the other works that we compare our results to, combination of features on multiple scales are used.

## 3.3 Convolutional Neural Networks for saliency prediction

The term 'neural network' has its origins in the attempts to find a good mathematical representations of information processing in biological systems. The first artificial neuron, the Threshold Logic Unit (TLU), was proposed by Warren McCulloch and Walter Pitts in 1943[4]. This type of neuron is depicted in Fig. 19 and it can be described by the following equations [4]:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j, \tag{12}$$

$$y_k = \varphi(u_k + b_k), \tag{13}$$

where $\varphi$ is a treshold function:

$$\varphi(v_k) = \begin{cases} 1 & if \quad v_k \geq 0, \\ 0 & if \quad v_k < 0. \end{cases} \tag{14}$$

A single-layer neural network that consist of such neurons is limited to the classification of linearly separable patterns. Regardless whether we use hard limiting or soft limiting as the source of nonlinearity in a single-layer model, it can classify only linearly separable patterns. Therefore, introducing more layers is necessary. If we do so, but we use a linear transfer function, the resulting network will be such that it can be equivalently represented by a single-layer network. Therefore, a non-linear function is necessary to gain the advantages of a multilayer network. To summarize, if we want to build a model that is able to learn more complex functions, we should consider a *multilayer* perceptron with the following properties [4]:

1. the model of each neuron in the network includes a nonlinear activation function (that is differentiable),

2. the network contains one or more layers that are hidden from both the input and the output nodes,

3. the network exhibits a high degree of connectivity, the extent of which is determined by synaptic weights of the network (we will see that this does not quite hold for the convolutional type of neural networks).

Figure 20 shows the architecture of a multilayer perceptron with two hidden layers and an output layer. This network is fully connected, i.e. every neuron in a network is connected to all the neurons in the previous layer. The hidden neurons act as feature detectors; as such, they play a critical role in the operation of a multilayer perceptron. As the learning process progresses across the multilayer perceptron, the hidden neurons begin to gradually discover the salient features that characterize the training data. They do so by performing a nonlinear transformation on the input data into a new space called the feature space. In this new space, the classes of interest in a pattern-classification task may be more easily separated from each other than in the original input data space.

2D Convolutional Neural Networks (CNNs) are a special class of a multilayer perceptron where instead of a fully connected layer, a 2D convolutional operation

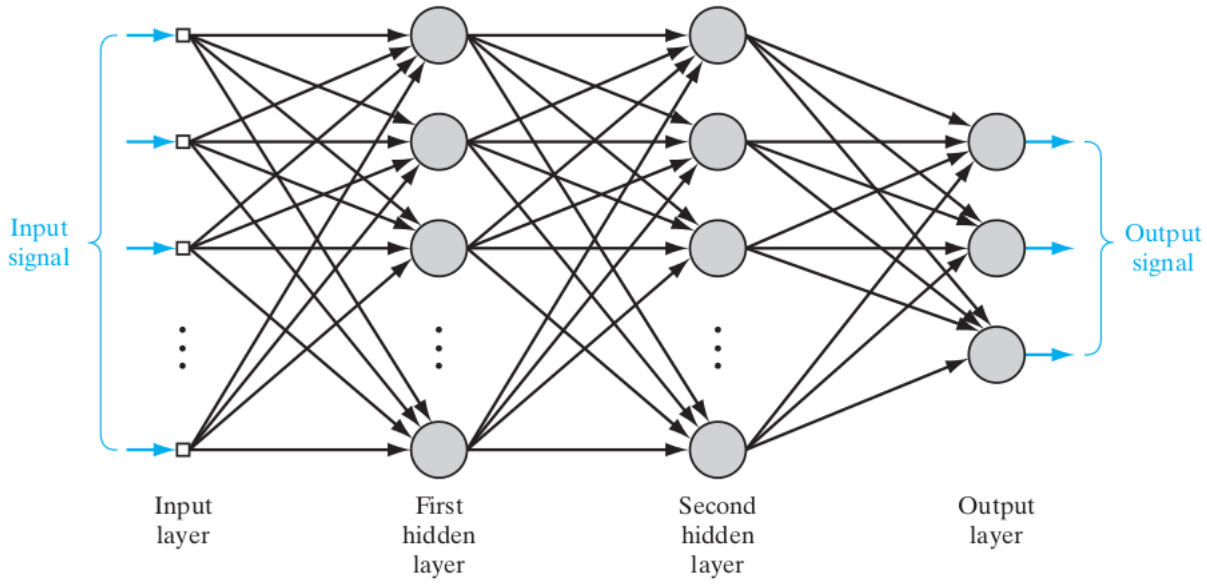



Figure 20: Multilayer perceptron with two hidden layers [4]

over an input image (or input maps when higher layers of a network are in question), is performed. Instead of applying equation (12) to obtain the output $y_k$, convolutional operator is applied using locally connected shared weights over the input in order to obtain a new map $\mathbf{u}^k$ of responses to local filters:

$$u_k^{ij} = (\mathbf{W} * \mathbf{x})^{ij}, \tag{15}$$

$$y_k^{ij} = \varphi(u_k^{ij} + b_k), \tag{16}$$

where $*$ is a convolutional operator, $i$ and $j$ are positions in the image (or a map when higher layers of a network are in question), $\mathbf{u}^k$ is the output map and $\mathbf{W}^k$ are the 2D weights of a $k$-th filter. By doing so, a couple of beneficial properties of the CNN architecture are achieved[9]:

- sparse connectivity

  CNNs exploit spatially local correlation by enforcing a local connectivity pattern

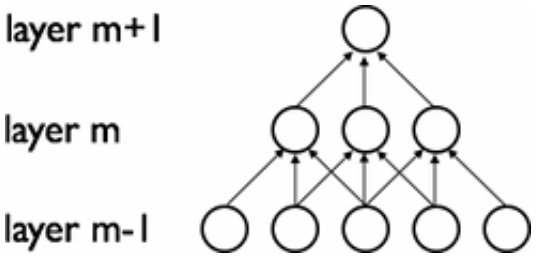



Figure 21: Illustration of increasing globality of features [9]

between neurons of adjacent layers, i.e. the input hidden units in the $m$-th layer are connected to a local subset of units in the $m-1$-th layer (see Fig. 21). If the layer $m-1$ is the input, then units in the layer above (layer $m$) have receptive fields of width 3 with respect to the input and are thus connected to only 3 adjacent neurons in the layer below (layer $m-1$). Units in layer $m+1$ have the same type of connectivity with the layer below. Their receptive field with respect to the layer below (layer $m$) is also 3, but their receptive field with respect to the input (layer $m-1$) is larger - it is 5. Since each unit is unresponsive to variations outside of its receptive field with respect to its input the architecture confines the learned filters to be local with respect to the the layer below. However, stacking many layers leads to the emergence of increasingly global filters. For example, the unit in hidden layer m+1 can encode a non-linear feature of width 5.

- shared weights

  In CNNs, the responses of the filters $\mathbf{W}$ are replicated across the entire visual field and they form a feature map, as shown in Fig. 22. Here, we show 3 hidden units belonging to the same feature map. Weights of the same color are shared, i.e. they are constrained to be identical. Replicating units in this way allows for features to be detected regardless of their position in the visual field. Additionally, weight sharing greatly reduces the number of free parameters to learn.

Once a feature has been detected, its exact location may be less important. Memorizing the exact location can actually be harmful because the position of a sin-

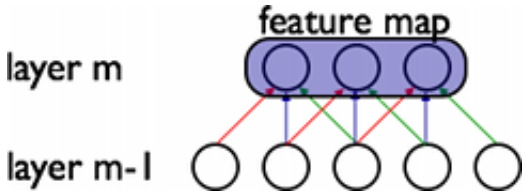



Figure 22: Feature map created using shared weights (shown in 1-D)[9]

gle pattern can vary to a certain degree for a different examples of the same class. Therefore, after a convolutional layer, a spatial pooling/subsampling is usually implemented. Pooling layers usually compute the max or average value of a particular feature over a predefined region of the input. A detailed theoretical analysis on those two types of pooling is given in [143]. Reducing the spatial resolution of the obtained feature maps is a way of reducing the precision with which position of a distinctive features are encoded in a feature map [69], i.e. it is a way of achieving *small* translational invariance. This also reduces the computational cost of subsequent convolution over the obtained maps.

In a typical neural network architecture, after a few convolutional and pooling layers, one or more fully connected layers are usually implemented (as in Fig. 20).

Finally, on the top most layer the softmax regression model is used as a classifier. This model generalizes logistic regression to classification problems where the class label can take on more than two possible values. Let $K$ be the number of classes and $\{\mathbf{w}_i, b_i\}_{i=1}^K$ the parameters of the logistic regression classifier. The corresponding cost of an example $\{\mathbf{x}, y\}$ is:

$$J(\{\mathbf{w}_i, b_i\}_{i=1}^K) = \sum_{j=1}^{K} 1\{y = j\} \log P(Y = j|\mathbf{x}, \{\mathbf{w}_i, b_i\}_{i=1}^K), \tag{17}$$

where

$$P(Y = j|\mathbf{x}, \{\mathbf{w}_i, b_i\}_{i=1}^K) = \frac{e^{\mathbf{w}_j \mathbf{x} + b_j}}{\sum_{k=1}^K e^{\mathbf{w}_k \mathbf{x} + b_k}}. \tag{18}$$

The total cost is the sum over all training examples.

After $\{\mathbf{w}_i, b_i\}_{i=1}^K$ are obtained, the class prediction for an example $\mathbf{x}$ is done in the

following way:

$$y_{pred} = \underset{j}{\operatorname{argmax}}\, P(Y = j|\mathbf{x}, \{\mathbf{w}_i, b_i\}_{i=1}^{K}). \tag{19}$$

There is no known closed-form way to solve for the minimum of $J(\{\mathbf{w}_i, b_i\}_{i=1}^{K})$ and therefore gradient descent is usually employed for solving this optimization problem. The rest of the parameters in the network architecture are obtained by applying the chain rule while using the same supervision criterion and gradient descent algorithm.

For the problem of saliency prediction, we adopt such fully supervised approach and use a 2D and 3D CNN The reasons for adopting this approach are presented in 2.1.2 and 2.1.4. Our network acts as a binary classifier that classifies cuboids as being fixations or not. The classifier we use is a logistic regression described in the previous paragraph. The input during the training are cuboids extracted around points fixated by humans (fixations) and cuboids that are at a distance twice as large as the size of a cuboid from the points fixated by humans (non fixations).

Our architecture for saliency prediction which classifies cuboids as fixations or non fixations is very similar to the one depicted in Fig. 4. Our implementation simplifies LeNet model[69] in the following ways:
- there are no bias parameters,
- pooling is by max operator (not average),
- hyperbolic tangent squashing function $\varphi$ (i.e. neurons output - see Eq. (16)) is modeled as: $y_k^{ij} = \tanh(u_k^{ij} + b_k)$, while [69] additionally implements a scaling factor,
- in the final layer we use softmax classifier rather than an RBF network, i.e. there are no Gaussian connections,
- there is only one (instead of two) fully connected MLPs on top of convolutional and pooling layers,
- convolutions at the second layer are fully connected, i.e. every filter in the second layer is applied on all feature maps from the first layer, while in [69] second layer filters are applied on a fixed subsets of maps from the first layer,
- in some experiments regarding spatiotemporal saliency prediction we have used 3D

Table 2: Parameters of 2D and 3D CNNs for saliency prediction

| Parameters | 2D | 3D |
|---|---|---|
| input cube dimensionality | 31x31 | 21x21x10 |
| size of 1st layer filters | 4x4 | 4x4x3 |
| 1st layer subsampling | 2x2 | 2x2x1 |
| number of 1st layer filters | 25 | 25 |
| size of 2nd layer filters | 2x2 | 2x2x3 |
| 2nd layer subsampling | 3x3 | 2x2x1 |
| number of 2nd layer filters | 50 | 50 |
| units in fully connected layer | 10 | 50 |
| batch size | 200 | 200 |
| learning rate | 0.01 | 0.05 |
| number of iterations | 20 | 30 |

instead of 2D convolutions, i.e. instead of Eq. (15) and Eq. (16), we have:

$$u_k^{ijt} = (\mathbf{W} * \mathbf{x})^{ijt}, \tag{20}$$

$$y_k^{ijt} = \varphi(u_k^{ijt} + b_k). \tag{21}$$

Here, we have additionally index $t$ which denotes a temporal position in the video cube, and and $\mathbf{W}^k$ are 3D weights of a $k$-th filter.

The rest of the parameters of the 2D and 3D CNN architectures we used in our experiments are listed in Table 2. The implementation is made using Theano framework [144, 145]. We have used small filter size in order to reduce the number of parameters and make the training easier, and a small number of units in the hidden layer in order to prevent overfitting. We have experimented with different number of filters and, as expected, found that the larger the number of filters the better (performance saturates for number of filters reported in Table 2). However, for 2D network we did not perform exhaustive experiments using different parameters and larger input size (the input patch size for tICA was larger, 41x41) as the point of those experiments was only to show that fully supervised network can easily outperform tICA architectures which are not trained using supervision criterion on all layers.

## 3.4 Experimental results

We validate our 2D saliency prediction methods on three publicly available datasets that contain images of natural scenes and corresponding recorded eye fixations. The datasets are described in 2.1.3.

In all our experiments we follow the same training and testing protocols as in the other literature that we compare our results to. In addition to that, for the MIT and Toronto datasets, we experimented with different sampling protocol, i.e. we differently defined areas for sampling fixations and non-fixations. In our case sampling fixations in 1% of most salient areas and non-fixations in lowest 70% of salient areas worked best, as opposed to 20/70 used in the other literature. Both results we obtained with 20/70 and 1/70 sampling protocol were better when compared to the ones from the literature. In the sampling protocol for the Kienzle dataset fixations are defined strictly as the points where humans have looked at, and we did not make further experiments regarding sampling for this dataset.

In 3.4.1 we will present saliency prediction results in image domain. First we will present the results obtained using tICA criterion on lower layers and an SVM on top of them. Second, we will present the results obtained using SVM supervision criteria in the (mid) pooling layer of tICA. Third, we will present the results obtained using fully supervised 2D convolutional neural network. In 3.4.2 we will present saliency prediction results in video domain. Here, we will investigate the importance of using 3D instead of 2D convolutions for spatiotemporal saliency prediction.

### 3.4.1 Image saliency prediction

#### 2D saliency prediction using tICA+SVM

Real valued saliency maps are obtained by computing per pixel SVM responses $\mathbf{w}^T\mathbf{x} + b$ where $\mathbf{x}$ are tICA features calculated on a patch centered around the point in question (see Eq. (3)). Then, as proposed in [36], the maps are smoothed with a Gaussian filter. Following the literature we report our results by means of the AUC (Area Under the ROC Curve). This is the most common score that is used to evaluate the results of saliency prediction. The ROC curve is drawn as the false positive rate vs. true positive rate by varying the threshold and using human fixations as a ground truth. The area under this curve indicates how well the saliency map predicts human

eye fixations.

In Table 3 we compare our results to the state of the art in the learning saliency paradigm without the central bias. AUC scores for the MIT and Toronto datasets are normalized, i.e. divided by the human agreement score.

In [37, 2] the feature vector consists of two color, one intensity, four orientation and a face channel. Our results outperform the ones using linear integration with optimal weights learned using linear, least square regression, even though they use as feature the output of a high level face detector. In general, the latter has shown to be an important feature for saliency prediction and in [2] it was found that the face channel is the most informative for the MIT dataset. We can see that the results obtained by [2] that uses nonlinear integration, i.e. an AdaBoost classifier, outperform ours. However, for the MIT and Toronto datasets our goal is to compare the performance of linear classifiers and handcrafted features and we have shown that when using a linear integration our features outperform handcrafted features.

On the Kienzle dataset the only reported results are for the model of Itti *et al.* [23] which uses the same channels but without learning their weights using human ground truth data, and for the model of [35] in which centre surround patterns are learned with an RBF SVM on raw image patches. Our model outperforms both and achieves state-of-the-art on this dataset, which is the most challenging dataset since it contains scenes taken in the nature that are without any particular regions of interest, as opposed to other two datasets. In the same table we show that our proposed joint optimization procedure (learned pooling) gives consistently better results than using linear SVM on top of tICA features - the improvements are 0.9%, 1.0% and 0.1% for MIT, Toronto and Kienzle datasets, respectively.

In order to give some insight to the advantages and limitations of the proposed method we illustrate some examples of our saliency maps in comparison to the human ground truth in Fig. 23. The maps of tICA are the ones obtained when using an original tICA algorithm with linear SVM on top of them, since there is not much visible difference in those maps and the ones obtained with supervised pooling. The maps were tresholded in a way that the 20% highest values in the map are considered salient.

In the first three rows we show some representative examples of our saliency maps to illustrate how our method can usually predict human fixations very well in scenes

Table 3: Comparison to the state-of-the-art. Our results marked with * are the ones obtained with 1/70 sampling.

| | MIT | Toronto | Kienzle |
|---|---|---|---|
| Linear integration [23, 39] | 0.776 | 0.828 | - |
| Linear integration with optimal weights [37] | 0.792 | 0.834 | - |
| Itti *et al.*[23] | - | 0.828 | 0.620 |
| Center-surround patterns [35] | - | - | 0.640 |
| Ours (tICA + linear SVM) | 0.790 | 0.841 | 0.654 |
| Ours (learned pooling) | 0.811 | 0.857 | **0.655** |
| Ours (2D CNN) | 0.823 | - | - |
| Ours* (tICA + linear SVM) | 0.803 | 0.850 | - |
| Ours* (learned pooling) | **0.812** | **0.860** | - |
| Ours* (2D CNN) | **0.823** | - | - |
| Nonlinear integration [2] | 0.876 | 0.916 | - |

that obtain clear shapes and objects like cars, buildings and traffic signs. However, we also show a few examples of saliency maps that do not match to the human ground truth for one of the following three reasons. First, the image in question may contain semantic content, for example letters (row 4) or faces (row 5). Second, the image in question may contain no distinctive areas of interest (rows 6, 7 and 8). Third reason is central bias (rows 6 and 7).

**2D saliency prediction using CNN**

When using a CNN, the saliency maps are constructed in the following way. Given an input patch extracted around pixel $p$, the softmax layer of a CNN gives as an output the probability of this patch being salient, that is:

$$s(p;\theta_s) = P(Y = +1|p,\theta_s), \tag{22}$$

where $\theta_s$ are the learned parameters of a CNN. This probability is used as a saliency value of pixel $p$. When applying 2D CNN for image saliency prediction, we calculate $s(p;\theta_s)$ for each point $p$ in the image.

The results of this model are also presented in Table 3. We have validated this model on the largest and most commonly used dataset, that is the MIT dataset. We

(a) (b) (c)

Figure 23: (a) stimuli image, (b) human ground truth saliency map, (c) our saliency map

Table 4: Saliency prediction results on the UCF sports dataset

| Parameters | AUC (test) | NCC (test) | AUC (all) | NCC (all) |
|---|---|---|---|---|
| 2D on patches | 0.8149 | 0.3050 | 0.8279 | 0.3231 |
| 2D on cubes | 0.8221 | 0.3170 | 0.8353 | 0.3375 |
| 3D on cubes | 0.8336 | 0.3512 | 0.8500 | 0.3734 |

can see that the results outperform both the results obtained when using only tICA at the lowest layer and the ones obtained using SVM supervision at the pooling layer.

### 3.4.2 Video saliency prediction

Constructing saliency maps for videos is a bit different than for images. For computational reasons, when applying 2D/3D CNNs to videos we do not calculate $s(p;\theta_s)$ for each pixel $p$. Those maps are obtained by sampling points across the video with a variable spatial step size: we sample a fixed number of points across spatial dimensions by varying strides across x axis and y axis of one frame. In practice, we sample 10 points across x axis and 20 points across y axis, and obtain saliency maps of dimensionality 10x20. Such maps are then resized to the original frame size.

In Table 4 we present saliency results on the UCF sports dataset. Here, we report the results using two measures. First we use AUC score, which we have used for the image saliency prediction as well. Second, we use Normalized Cross Correlation, as the only saliency results for UCF sports dataset are reported in [146] using this measure. We have trained our saliency prediction networks using a fixed training set that is defined by the train/test split introduced in [104] for the problem of action recognition. As [146] report the results on the whole dataset, we report the saliency prediction results both on the test split and on the whole dataset.

Here, we would like to illustrate how different architectural properties regarding temporal processing affect the saliency prediction results. For that purpose we have trained three different CNNs. The first network (2D on patches) acts in the same way as the one for images: it takes as an input 2D patches around points and performs 2D convolutions on them and on the maps in the higher layer. The second network (2D on cuboids) takes as an input 3D cuboids around points and performs 2D convolutions on them and on the maps in the higher layer. Finally, the third network (3D on

cuboids) takes as an input 3D cuboids around points and performs 3D convolutions on them and on the maps in the higher layer.

As expected, the best results are consistently obtained for the third network (3D on cubes) and the worst for the first network (2D on patches). However, we would like to emphasize that the input to the first network contains 10 times less information than the inputs to the second and third networks (as the temporal length of a cuboid is 10, i.e. we take 10 consecutive frames to construct an input cuboid). Therefore, it is interesting to notice that larger improvements in the results are obtained when comparing the third and the second network, than when comparing the second and the first, even though the third and the second are applied on the same input. Some examples of saliency prediction maps will be shown in 4.2.3. In comparison to the state-of-the-art, the very recent work of [146] reports 0.47 using NCC measure, which outperforms our results. This is due to the fact that at the higher levels of their approach more sophisticated manually designed methods are used. Note that we cannot compare our video saliency prediction results to the works that are closest to ours, that is [74, 43], for the following reasons. In [43] saliency prediction results are reported only for randomly selected frames from Hollywood2 dataset. In [74] saliency prediction results are reported only for the areas across bounding boxes inferred by the proposed method.

## 3.5 Conclusions

In this chapter we have proposed few methods that learn bottom-up visual saliency by learning features instead of using manually selected ones. We have shown that the performance of our model using only these learned low level features gave better results than more sophisticated approaches where higher level features such as face detectors were used (on MIT and Toronto dataset). We have also achieved state of the art results on Kienzle dataset.

Our low layer features were first learned in an unsupervised manner and the SVM supervision criterion was used as a classifier only at the top-most layer of the whole architecture. As our goal was to investigate the impact of incorporating the supervision criterion on lower layers we have developed an optimization scheme that optimizes the tICA pooling matrix weights jointly with SVM weights. The improvement in the

results when using such optimization scheme are consistent with the work of [61], where the importance of fine-tuning the network according to supervision criterion is emphasized.

In the next step we wanted to use an architecture that will be trained using only supervision criterion on all layers. Additionally, for the reasons presented in 2.1.1 and in 2.1.2 we wanted to change some of the architectural properties, that is to use deeper architecture and to incorporate the convolutional operator. Therefore, we have used fully supervised deep convolutional neural networks and shown that it outperformed the results of the other architectures which do not use convolutions and supervision criterion on all layers.

In addition to that, we have investigated how some changes in the architectural properties affect the saliency prediction in spatiotemporal domain. We have shown that extending spatial convolutions to spatiotemporal consistently yields better results. This suggest that the motion is important for saliency prediction.

To summarize, we have shown that there are gains in the performance by changing certain architectural properties and incorporating supervision in more layers. In the following chapter we will show how 3D CNN for saliency prediction we have described here is used in a couple of action recognition frameworks. In addition to that, in 4.1.1 we will describe how we develop a similar 3D CNN for action feature extraction.

# 4 Human action recognition using saliency learned from recorded human eye fixations

Detecting interesting/salient regions in a video is very helpful for the problem of action recognition due to the fact that videos are in most cases weakly labeled, i.e. only the class of the whole unsegmented video is given. Both during training and testing, it is not known where within a video the action takes place. Obtaining manual localizations of the actions is a very cumbersome and time-consuming process. Identifying the salient parts of the videos is the first issue we address in this chapter. In order to do so, we will use our saliency prediction method for videos presented in 3.3.

The second issue we address is related to the fact that videos usually contain occlusions, illumination and viewpoint changes, jitter caused by camera movements and other types of noise. Hence, it is difficult to choose appropriate features that are able to deal with these variations that can occur in the data. We will learn features by training a supervised 3D convolutional neural network that extracts compact features on local cuboids. Given that humans tend to look at the important and discriminative parts of the action video[43, 18], we will train our network only on cuboids extracted around recorded gaze fixations.

In this chapter we will present a couple of action recognition frameworks that use saliency learned from human eye fixations and show that using saliency improves the action classification results. The recorded gaze fixations will be used when learning saliency and action features, both using 3D CNN. Learned saliency and action features will be then used in the action recognition frameworks. Figure 24 shows an overview of the inference procedure of our proposed frameworks for a single video.

The first phase is the same for both frameworks (PHASE 1 in Figure 24). We use the outputs of two separate 3D CNNs - one for saliency prediction (addressing the issue of finding relevant discriminative parts of the videos) and the other trained to extract discriminative action features (addressing the problem of finding good features).

In order to show that using saliency learned from recorded human gaze alleviates the problem of action recognition we will employ our 3D CNN trained to discriminate

between fixated and non fixated points in two frameworks: first in the majority voting framework (PHASE 2A in Figure 24) where we also show a good discriminatory power of our learned action features, and second in the SVM framework (PHASE 2B in Figure 24). In the PHASE 2B we will use an SVM model, which as an input feature representation uses the features obtained by a 3D CNN that was trained to learn action features.

As it can be seen in Figure 24, during inference we are using the outputs of two 3D CNNs to construct an input to the SVM classifier that predicts the video class and the bonding boxes at each frame. The first 3D CNN predicts saliency and the second 3D CNN extracts action features. Recorded human fixations are required for training both of those networks.

In our proposed SVM model we are introducing the latent variables that indicate where the action takes place. Those are the locations of the bounding boxes and they are unknown during both training and testing. During testing/inference our method optimizes with respect to both the video label and the location of the bounding boxes a cost that comprises of two terms. First, a classical SVM term and second, the saliency within the bounding box. Our SVM tries to find the bounding box in a way that it balances between good feature response across the bounding box area and a high saliency concentration within the same bounding box. The optimization scheme we developed for this type of SVM shows improvements in action recognition results over the baseline SVM that does not use bounding boxes, over the baseline SVM that uses the most salient bounding boxes and over the latent SVM introduced in [1]. Furthermore, we show that using saliency yields significant improvements in our proposed SVM framework.

The main contributions of this chapter are the following:

- we show the usefulness of our learned saliency prediction in a majority voting and an SVM framework,
- we present a fully supervised method for learning action features using human gaze information and show that those features outperform commonly used handcrafted features in a majority voting framework,
- we present a latent SVM based method for joint action recognition and localization in which the class label and the bounding boxes are inferred by optimizing

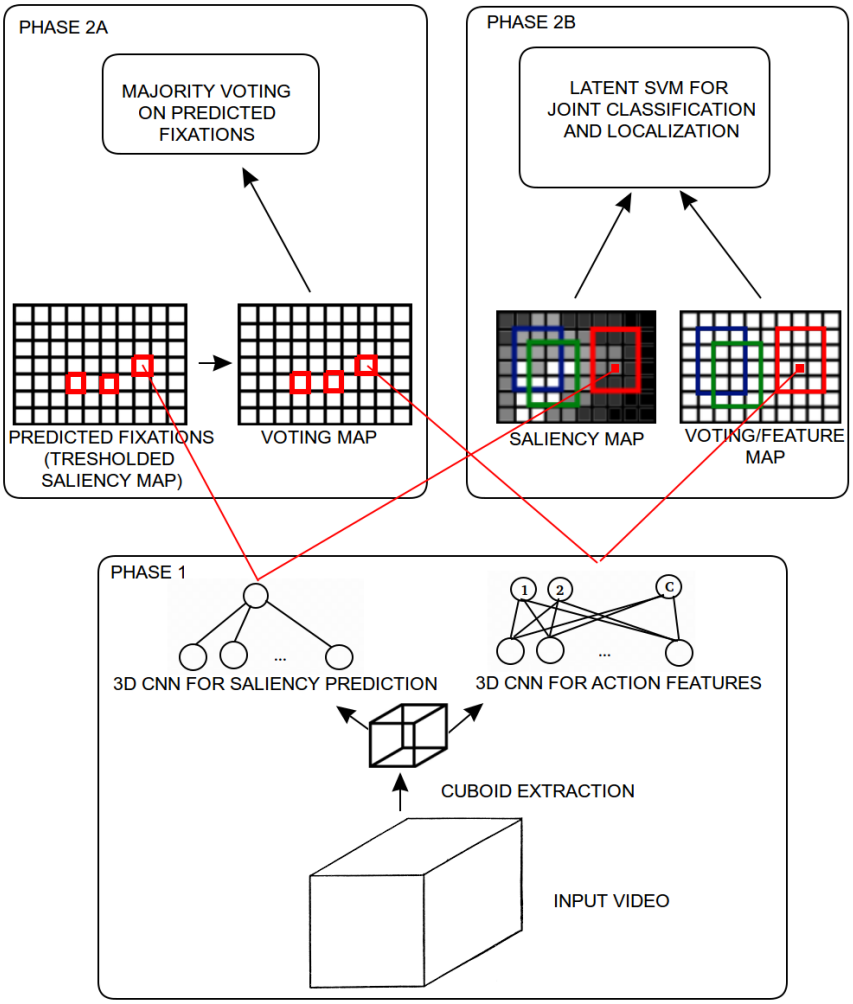



Figure 24: Illustration of two phases in action class inference procedure for the whole video

a cost function that contains a missclassification penalty term and a term that is related to the saliency within the bounding box. We show that our joint localization and recognition model that uses predicted saliency for bounding box inference is better than both the model that does not use predicted saliency and the model of [1].

## 4.1 Majority Voting for action recognition

We will present a simple majority voting framework for action recognition to show two things. First, we would like to show the efficacy of our learned local discriminative action features. We will do so by employing them in this framework and comparing them to the frameworks where handcrafted features are used. Second, we would like to show how learned and ground truth saliency can be used in this framework and improve the action recognition results.

Note that recorded gaze fixations are necessary both for learning action features and for learning the saliency. When learning action features the training data will be selected based on the gaze fixations. When learning saliency fixations and non fixations are used as positive and negative examples for training the classifier (as described in 3.3).

In 4.1.1 we will present how we learn local action features and use them in a majority voting framework. In 4.1.2 we will present the results we obtain in a majority voting framework that uses learned saliency and learned action features. In 4.1.3 we will present the conclusions.

### 4.1.1 3D Convolutional Neural Network for action features

The network that learns action features acts as an action classifier on a local cuboid level. During training, the input to the network for learning action features are cuboids that are extracted only around points at which humans fixate, and the output is the class of the cube. By training this network only on fixated points, we learn discriminative features discarding the background clutter.

The architecture used in our experiments is depicted in Figure 25. This architecture has 10 outputs in the last layer, one for each of the 10 action classes. This network is very similar to the one we used in 3.3 for 3D saliency prediction (see Fig. 4), the only difference being that the former had only two outputs in the last layer and fewer number of filters in both layers. There are also some differences in the architectural parameter selection, which are listed in Table 7. The implementation is made using Theano framework [144, 145].

When classifying the whole video sequence we are extracting a fixed number of cuboids across the video and deciding on the class of the video based on the majority

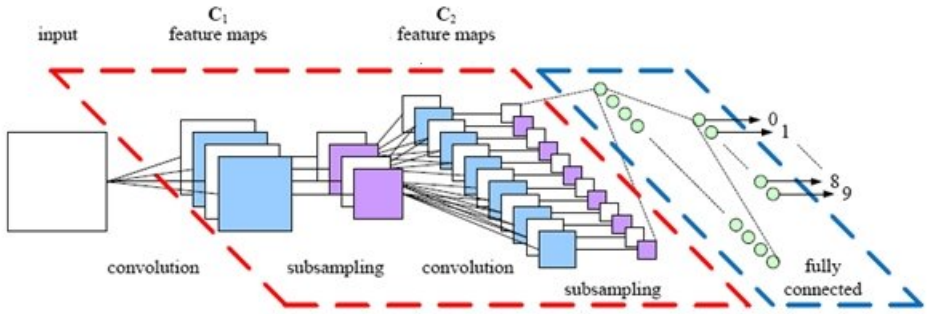



Figure 25: Convolutional neural network architecture used in our experiments for action features learning. Here, input and maps are depicted as 2D, in practice they are 3D.

Table 5: Parameters of 3D-CNN for action features

| Parameters | values |
|---|---|
| input cube dimensionality | 21x21x10 |
| size of 1st layer filters | 4x4x3 |
| 1st layer subsampling | 2x2x1 |
| number of 1st layer filters | 50 |
| size of 2nd layer filters | 2x2x3 |
| 2nd layer subsampling | 2x2x1 |
| number of 2nd layer filters | 100 |
| units in fully connected layer | 50 |
| batch size | 200 |
| learning rate | 0.05 |
| number of iterations | 30 |

of the cuboid votes. The vote of a single cuboid extracted around point $p$ is:

$$c^* = \underset{c}{\operatorname{argmax}}\, \mathbf{a}_c(p;\theta_a) = \underset{c}{\operatorname{argmax}}\, P(Y = c|p,\theta_a) \tag{23}$$

where $c$ is an action class and $\theta_a$ are the learned weights of the 3D CNN for action features.

In our approach the votes of the cuboids are sampled across the whole video in three different ways: first without using saliency, i.e. randomly, second using saliency

predicted by our 3D CNN network trained for saliency prediction and third using ground truth saliency. In the setup where ground truth saliency is used, we simply use only the votes of the recorded fixations points. In the setup with predicted saliency we are using the output of a pretrained saliency predictor (3D-CNN described in 3.5) which gives as output the probability of a cuboid being salient, that is $s(p;\theta_s) = P(Y = +1|p, \theta_s)$, where $\theta_s$ are the learned weights of 3D CNN for saliency prediction. A cuboid is classified as a salient if $s(p;\theta_s) \geq 0.5$ and in that case we count its vote, otherwise we discard it. After counting the votes, the class with the most votes wins.

### 4.1.2 Experimental results

We evaluate our method on the UCF sports dataset [137] using human eye movements recorded for each video of this dataset [43]. Some works that validate their methods on the UCF sports dataset use leave-one-out protocol, however, we follow the one from [74]. In this protocol the dataset is split in 103 training examples and 47 test examples. The exact training-testing split is available at http://www.sfu.ca/ tla58. We follow this protocol for a couple of reasons. First, this is the work closest to ours. Second, in [104], where this protocol was introduced, it has been shown that there is a strong scene correlation among videos in certain classes. This might cause a system to learn scene correlations rather than action correlations. Third, in some works that use LOO cross validation the parameter setting is unclear [104].

In Table 6 we present the results obtained by a simple majority voting scheme in order to illustrate two things. First, in order to show a good discriminatory power of the features learned with CNN we compare the results obtained by a simple majority voting scheme to the ones obtained with the BoW approach. In the BoW approach cuboids are densely sampled [2] and HoG, HoF and HOMB descriptors are used - for more details see [74].

We can see that our simple majority voting scheme that uses only learned features, without the additional quantization step and training an SVM classifier as it is done in the BoW approach, yields much better results, both when using predicted saliency and when using ground truth saliency. Even when using no saliency, the result is comparable with the global BoW setting. This shows that our discriminatively

[2]Interestingly, in realistic videos, dense sampling has been shown to be a better sampling strategy than using any kind of STIP[79, 18].

| Method | MAP |
|---|---|
| Global BoW [74] | 64.29 |
| BoW with spatial split [74] | 65.95 |
| BoW with temporal split [74] | 69.64 |
| Majority Voting (no saliency) | 64.31 |
| Majority Voting (with predicted saliency) | 74.17 |
| Majority Voting (with ground truth saliency) | 85.00 |

Table 6: Results obtained using majority voting scheme. The measure is mean per class classification accuracy.

learned features compare well to the handcrafted ones.

We can see that when using ground truth saliency, the result is 85.00%, which is around 3% above the results reported in [74] and around 1.5% above our best result obtained with SVM-based approach. Those results are reported in Table 8 and further discussed in section 4.2.3. Note that the results reported in Table 8 are obtained by frameworks in which either no saliency or only the predicted saliency is used during testing. There are no works that report the results using ground truth fixations.

However, even when the cuboids are sampled around the ground truth fixations, the classification scheme that we have just described is not sufficient to achieve 100% accuracy. This may be attributed, to different degrees, first to the extracted local features, and second to the voting scheme that is used to combine local information. Our method does not utilize geometric constraints and, may be confused by information at fixation areas that are common between different actions, such as at faces.

The second thing we want to illustrate in this majority voting scheme is the fact that there is a large improvement in results when using any kind of saliency prediction, either ground truth or predicted, over the results without saliency prediction, i.e. over the results obtained by random dense sampling. As mentioned, the latter were shown to be superior over sampling the points detected with any of the commonly used STIPs[79, 18]. This shows the efficacy of our 3D CNN-based saliency predictor.

In Table 7 we present results we obtain by varying the hyperparameters of 3D CNN for action recognition. We varied the number of filters in both layers, the number of fully connected units and the depth of the 3D CNN for action features. The size

| Network | 1st layer filters | 2nd layer filters | fully connected units | result |
|---|---|---|---|---|
| 0 | 50 | 100 | 50 | 85.11 |
| 1 | 50 | 50 | 50 | 85.11 |
| 2 | 25 | 100 | 50 | 76.60 |
| 3 | 50 | 100 | 25 | 78.72 |
| 4 | 50 | 100 | 100 | 78.72 |
| 5 | 25 | 50 | 50 | 78.72 |
| 6 | 50 | 50 | 25 | 78.72 |
| 7 | 50 | - | 50 | 74.47 |
| 8 | 50 | - | 25 | 72.34 |
| 9 | 10 | - | 50 | 68.09 |
| 10 | 50 | - | 50 | 68.09 |

Table 7: Results obtained in a majority voting framework when neural network for action features is learned using different hyperparamters. Here, as saliency prediction ground truth fixations are used. The measure is classification accuracy per video, as opposed to the mean per class classification accuracy reported in table 6.

of the filters is the same for all networks, that is 4x4x3 in the first layer and 2x2x3 in the second layer. The only exception is network 10, which has larger first layer filters, that is 8x8x5. We observe that even in one layer network training with larger filter size is challenging. Other things we observe are as follows. First, we can see that adding a layer improves the results. Second, larger number of filters is beneficial (in our case especially first layer filters - compare networks 0 and 1). Third, larger number of units in a fully connected layer leads to overfitting (network 4). Those observations verify general findings in the deep learning literature.However, we did not investigate the impact that different hyperparameters would have in the SVM framework: in further SVM experiments we use the largest network, that is network 0.

In Figure 26 we can see some examples of well estimated saliency maps and voting maps. Those maps are obtained by sampling points across the video with a variable step size - this will be described in more details in 4.2.1. For each point, its saliency value (see Eq. (29)) and voting vector (see eq. (26)) are obtained. Correct votes in the voting maps are the ones for which $c^*$ (see Eq. (23)) corresponds to the ground truth and those are marked with white. The incorrect ones, i.e. the ones that cast

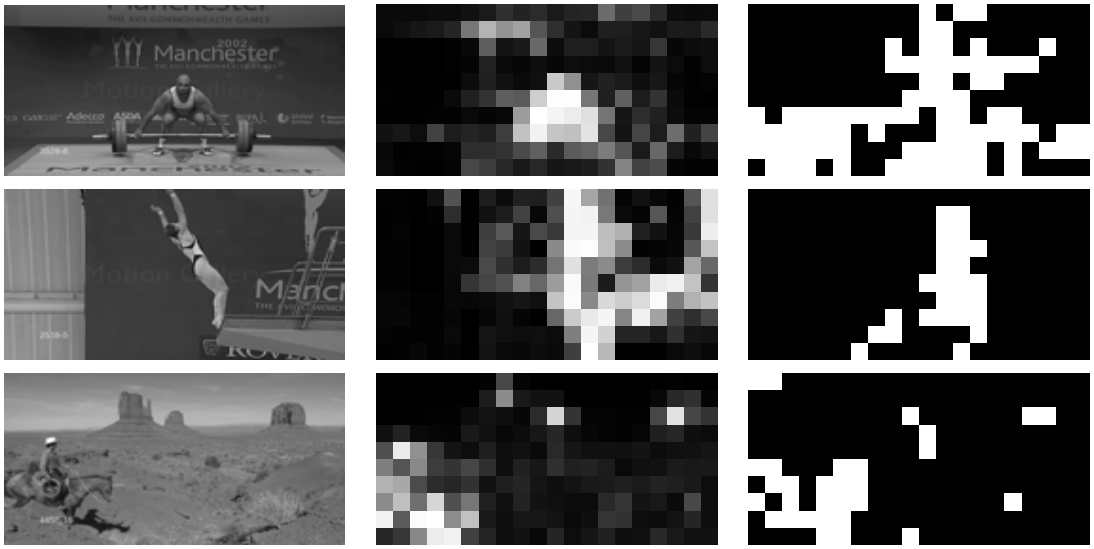



Figure 26: Examples of good saliency prediction and voting maps: (a) original frame, (b) predicted saliency map, (c) voting map.

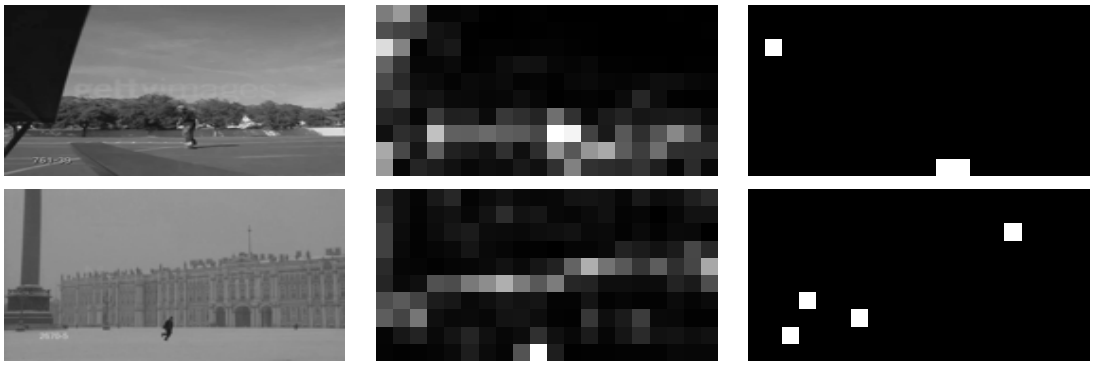


Figure 27: Examples of bad voting maps: (a) original frame, (b) predicted saliency map, (c) voting map.

vote for any other action than the correct one, are marked with black. In Figure 27 we see examples of misclassification of a video action class, and we notice that the misclassification is mostly due to errors in the voting scheme rather than in the saliency prediction. Those videos exhibit large change in scale and we can see that in those videos it is hard to notice the movements even with bare eye.

### 4.1.3 Conclusions

In this section we have presented a simple majority voting framework that uses only two separate 3D CNNs trained using recorded human gaze fixations. We have shown how recorded human fixations can alleviate the problem of action recognition: first, by using them for training a 3D CNN saliency predictor, and second, by using them for training a 3D CNN discriminative mid level feature extractor. The efficacy of our 3D CNN for action features was shown by using only the cuboid votes obtained by this network for the classification of the whole videos. The results obtained by this simple voting scheme compared to the state-of-the-art approaches. The efficacy of our 3D CNN saliency predictor was shown by using its output as a criterion for discarding the cuboid votes. When used in such way the saliency predictor improved the action recognition results significantly.

In the next section we will build up on this work and use the outputs of 3D CNN for action recognition and 3D CNN for saliency prediction in an SVM framework.

## 4.2 Support Vector Machine for action recognition

In 4.2.1 we will present the proposed SVM-based classifier. First, we will present how learned action features are used to build a representation that is fed in this classifier. Second, we will present how we define the saliency cost using the output of saliency prediction network and how this saliency cost is incorporated in the total SVM cost. Third, we will present the optimization procedure for this type of SVM. Finally, we will present how the classification of a video is performed. In 4.2.2 we will present how a special case of our model compares to the one of [1]. In 4.2.3 we will present the results of action recognition on two sports action datasets. In 4.2.4 we will present conclusions.

### 4.2.1 Support Vector Machine with added saliency cost

In this SVM framework we will show how predicted saliency can be used in order to infer the location of bounding boxes that contain relevant parts of the videos. In our SVM formulation, the positions of bounding boxes that capture the actions are included as additional parameters over which we optimize the SVM cost. That means that if we want to infer the class of a video (either training or testing) we have to search for a bounding box whose corresponding feature response is high for a certain class, which is the basic principle of a standard binary SVM: high feature response refers to a positive class, and low to a negative class. However, besides the corresponding feature representation, in our SVM each bounding box has its saliency cost.

The global representation of a whole video that is fed in our SVM framework consist of action features extracted by the 3D CNN across the fixed area around the inferred center of a bounding box. The high response of a bounding box can be due to background noise, so before inferring the class (of a bounding box) as positive we would want the area of that bounding box to have high saliency. The saliency of a bounding box is predicted using the output of our 3D CNN saliency predictor across the bounding box area. The idea is to add this saliency as a cost in a way that it would encourage the SVM to chose bounding boxes with high saliency rather than the ones with low saliency. However, there might be cases when a bounding box is salient while its content is not relevant for action recognition. For example, the bounding

box might capture the head of a person, or the wrong person. Such bounding box might have high saliency and low feature response.

Therefore, by searching for a bounding box with added saliency cost, we are trying to balance the effects of high saliency and good feature matching, as only one of those two cannot always be a good indicator of the action class. This will also be shown in the experimental results, as we will present the results when only one of those effects it taken into account (those will be two special cases of our SVM).

**Building a global feature representation**

Videos are in general of various length and resolutions, but as an input to our SVM, we need to build a video representation of a fixed dimensionality. In order to do so, we sample a fixed number of frames (i.e. we vary the stride for each video), and we sample a fixed number of points across spatial dimensions by varying strides across x axis and y axis of one frame. Action features and saliency are then extracted for all points sampled in such way across the whole video. The extracted action features are used to build a representation that is fed into an SVM. This is done simply by doing feature concatenation spatially across a bounding box of a fixed size, and then doing feature concatenation temporally, across a fixed number of frames.

Formally, the representation of a video $\mathbf{x}^i$, denoted by $\mathbf{R}(\mathbf{x}^i, \mathbf{bb}^i)$ , is a concatenation of features extracted at bounding boxes per frame $\mathbf{bb}^i = [bb_1 \ ... \ bb_t \ ...]^T$, $t = 1, \ ..., \ nt$ where $nt$ is the number of frames. That is, $\mathbf{R}(\mathbf{x}^i, \mathbf{bb}^i)$ consists of the concatenated features $\mathbf{r}(\mathbf{x}_t^i, bb_t^i)$ for each frame $t$. Formally:

$$\mathbf{R}(\mathbf{x}^i, \mathbf{bb}^i) = [\mathbf{r}(\mathbf{x}_1^i, bb_1^i)^T \ \ ... \ \ \mathbf{r}(\mathbf{x}_t^i, bb_t^i)^T \ \ ...]^T, \ t = 1, \ ..., \ nt, \tag{24}$$

where $\mathbf{r}(\mathbf{x}_t^i, bb_t^i)$ is the concatenation of features extracted by the pretrained 3D-CNN within the bounding box $bb_t^i$ in frame $t$, that is:

$$\mathbf{r}(\mathbf{x}_t^i, bb_t^i) = [... \ \mathbf{a}(p; \theta_a)^T \ ...]^T, \quad p \in bb_t^i, \tag{25}$$

where $p$ is a point within the predicted bounding box and $\mathbf{a}(p; \theta_a)$ is a vector of features extracted at point $p$ using the 3D CNN (with parameters $\theta_a$) trained for action cube classification. The feature vector $\mathbf{a}(p; \theta_a)$ is the output of the softmax layer of the 3D CNN and contains the probabilities that the cuboid $p$ belongs to each

of the class $c$. That is:

$$\mathbf{a}(p;\theta_a) = [P(Y=1|p,\theta_a),\ ...,\ P(Y=c|p,\theta_a),\ ...,\ ]^T, \tag{26}$$

where $c$ is an action class. Clearly, the dimensionality of $\mathbf{a}(p;\theta_a)$ is equal to the number of classes.

**Defining the saliency cost**

In our SVM framework we want to avoid choosing bounding boxes $\mathbf{bb}^i$ with low concentration of saliency. Hence, for each video $\mathbf{x}^i$, we add a cost which is defined in terms of the saliency concentration in the inferred bounding boxes. The saliency concentration within the bounding box $\mathbf{bb}^i$ for the video $\mathbf{x}^i$ is defined as:

$$M(\mathbf{x}^i, \mathbf{bb}^i) = \sum_{t=1}^{nt} m(\mathbf{x}_t^i, bb_t^i), \tag{27}$$

where $m(\mathbf{x}_t^i, bb_t^i)$ is the saliency concentration at bounding box $bb_t^i$ at frame $t$ defined as:

$$m(\mathbf{x}_t^i, bb_t^i) = \frac{\sum_{p \in bb_t^i} s(p;\theta_s)}{\sum_{p \in bb_t^i} 1}, \tag{28}$$

where $s(p;\theta_s)$ is the estimated saliency of a point $p$ using the parameters $\theta_s$ of the 3D CNN that is trained for saliency prediction. That is:

$$s(p;\theta_s) = P(Y=+1|p,\theta_s), \tag{29}$$

where $P(Y=+1|p,\theta_s)$ is the output of the softmax layer of the network for saliency prediction, that is the probability of a point $p$ being a fixation.

**Defining the total SVM cost**

Here, we define the SVM learning problem formally. Let $\mathbf{x}^i$ be a video of $nt$ frames and $\mathbf{bb}^i = [bb_1^i,\ ...,\ bb_t^i,\ ...,\ bb_{nt}^i]^T$ bounding boxes per frame that ideally contain discriminative information for action classification. Let $\mathbf{R}(\mathbf{x}^i, \mathbf{bb}^i)$ be a representation of the video in question, as described previously. Typical systems, such as [113, 119,

120], assume that the information $\mathbf{bb}^i$ is given, and adopt a video classification scheme such as $\mathbf{w}^T\mathbf{R}(\mathbf{x}^i, \mathbf{bb}^i) + b$, where $\mathbf{w}, b$ are learned using, for example, max-margin learning. For example, $\mathbf{bb}^i$ can be given in a form of a STIP detector that is used in order to sample the cuboids around salient points. The representation $\mathbf{R}(\mathbf{x}^i, \mathbf{bb}^i)$ that is built using those points is then fed into an SVM classifier and only SVM parameters are learned. By contrast, we treat the locations of the bounding boxes $\mathbf{bb}^i$ as latent variables and solve the optimization problem in which we are searching not only for the optimal values of the standard SVM parameters, but also for the optimal locations of the bounding boxes.

Given a set of labeled videos $D = \{(\mathbf{x}_1, y_1),\ ...,\ (\mathbf{x}_M, y_M)\}$ where $y_i \in \{-1, 1\}$ and $M$ is the number of videos, we are solving the following optimization problem:

$$\min_{\mathbf{w}, b, \{\mathbf{bb}^i\}_{i=1}^M} L_D(\mathbf{w}, b, \{\mathbf{bb}^i\}_{i=1}^M), \tag{30}$$

where

$$L_D(\mathbf{w}, b, \{\mathbf{bb}^i\}_{i=1}^M) =$$

$$\frac{1}{2}(\mathbf{w}^T\mathbf{w} + b^2) + \sum_{i=1}^{M} \big[\max(0, 1 - y^i f(\mathbf{w}, b; \mathbf{x}^i, \mathbf{bb}^i)) - \lambda M(\mathbf{x}^i, \mathbf{bb}^i)\big]. \tag{31}$$

In the above equation $f(\mathbf{w}, b; \mathbf{x}^i, \mathbf{bb}^i)$ is the scoring function for a video $\mathbf{x}^i$:

$$f(\mathbf{w}, b; \mathbf{x}^i, \mathbf{bb}^i) = \mathbf{w}^T\mathbf{R}(\mathbf{x}^i, \mathbf{bb}^i) + b, \tag{32}$$

and $\mathbf{w}$ are concatenated weights per frame:

$$\mathbf{w} = [\mathbf{w}_1^T,\ \ ...,\ \ \mathbf{w}_t^T,\ \ ...,\ \mathbf{w}_{nt}^T]^T. \tag{33}$$

Note that the additional cost $M(\mathbf{x}^i, \mathbf{bb}^i)$ related to the saliency of a bounding box areas is regularized by the parameter $\lambda$.

In Fig. 28 we illustrate the working of our proposed SVM, that is the selection of the bounding box and building of the input feature representation based on learned saliency and action features.

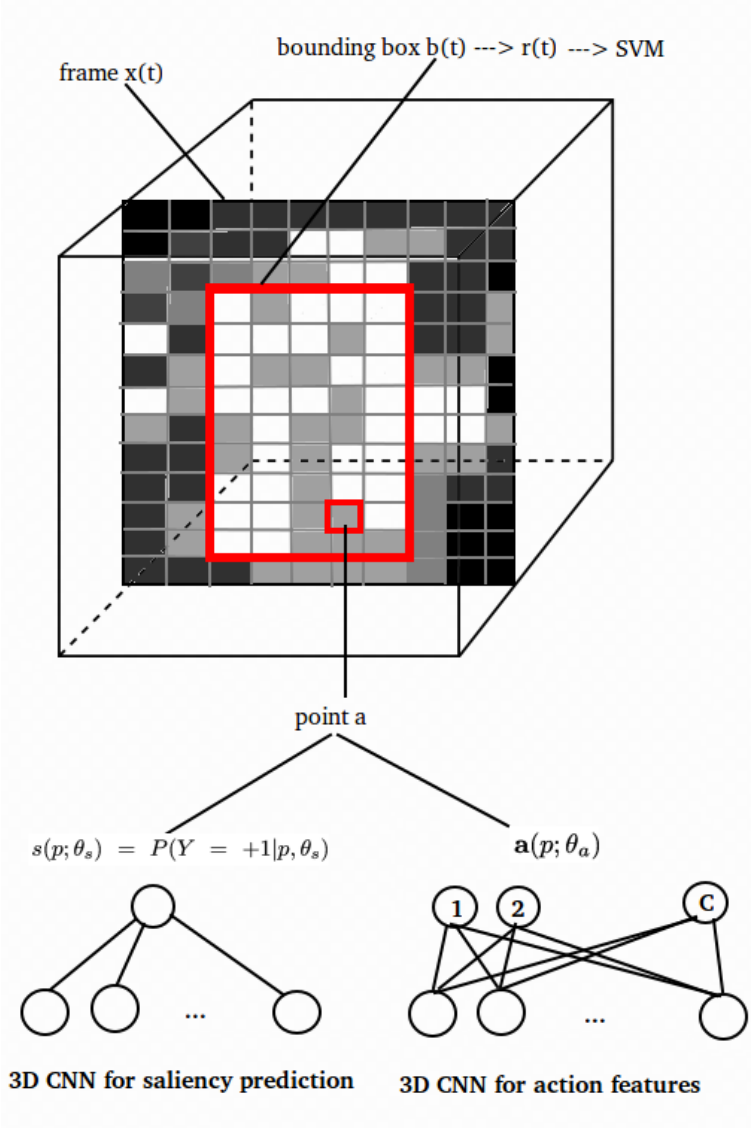



Figure 28: Illustration of the proposed SVM. In each frame $\mathbf{x}_t$ bounding box $bb_t$ is selected based on saliency concentration and features extracted across that bounding box. Based on the selected bounding box a frame representation $\mathbf{r}_t(\mathbf{x}_t, bb_t)$ is built by concatenating features across the bounding box. A video representation $\mathbf{R}(\mathbf{x}, \mathbf{bb})$ is further built by concatenating frame representations.

### Optimization

In order to minimize the cost function $L_D(\mathbf{w}, b, \{\mathbf{bb}^i\}_{i=1}^M)$ over two subsets of the parameters, namely $\mathbf{w}$ and $b$ on the one hand and $\{\mathbf{bb}^i\}_{i=1}^M$ on the other, we use a

block coordinate descent method. We iteratively alternate between optimizing the cost function with respect to the SVM parameters $\mathbf{w}, b$ keeping the bounding box parameters $\{\mathbf{bb}^i\}_{i=1}^M$ fixed (step 2) and optimizing the cost function with respect to the bounding box parameters $\{\mathbf{bb}^i\}_{i=1}^M$ keeping the SVM parameters $\mathbf{w}, b$ fixed (step 3). Step 2 results to a convex optimization problem, more specifically an SVM-like problem that we solve with a gradient descent method. Step 3 is an optimization problem that can be solved by enumeration of the positions of the $\{\mathbf{bb}^i\}_{i=1}^M$ - an efficient exact solution is possible given that we do not consider interdependencies in subsequent frames (see Eq. (41) - Eq. (45)). Therefore, each step gives optimal solutions with respect to the subset of the parameters and the procedure converges to a local minimum.

The full procedure consists of the following steps:

**Step 1. Initialization**

We initialize the bounding box $(bb_t^i)^*$ at frame $t$ as the one that maximizes the saliency:

$$(bb_t^i)^* = \underset{bb_t^i}{\operatorname{argmax}}\, m(\mathbf{x}_t^i, bb_t^i), \tag{34}$$

i.e. we are choosing the most salient areas of the videos. Note that this solution is actually the solution of a special case of our model when in the objective function the parameter $\lambda = +\infty$ (see Eq. (31)).

**Step 2. Optimization with respect to $\mathbf{w}, b$**

In this step we solve for $\mathbf{w}, b$ while keeping $\{\mathbf{bb}^i\}_{i=1}^M$ fixed. This results in a convex optimization problem that we solve by stochastic gradient descent. The subgradient of the objective function with respect to $\mathbf{w}$ is computed as follows:

$$\bigtriangledown_w L_D(\mathbf{w}, b, \{\mathbf{bb}^i\}_{i=1}^M) = \mathbf{w} + C \sum_i h_{\mathbf{w}}(\mathbf{w}, \mathbf{x}_i, y_i), \tag{35}$$

where

$$h_{\mathbf{w}}(\mathbf{w}, \mathbf{x}_i, y_i) = \begin{cases} 0, & \text{if} \quad y_i f(\mathbf{w}, b; \mathbf{x}^i, (\mathbf{bb}^i)^*) \geq 1, \\ y_i \mathbf{R}(\mathbf{x}_i, (\mathbf{bb}_i)^*), & \text{otherwise.} \end{cases} \tag{36}$$

and the subgradient with respect to $b$:

$$\bigtriangledown_b L_D(\mathbf{w}, b, \{\mathbf{bb}^i\}_{i=1}^M) = b + C \sum_i h_b(b, \mathbf{x}_i, y_i), \tag{37}$$

where

$$h_b(b, \mathbf{x}_i, y_i) = \begin{cases} 0, & \text{if} \quad y_i f(\mathbf{w}, b; \mathbf{x}^i, (\mathbf{bb}^i)^*) \geq 1, \\ y_i, & \text{otherwise.} \end{cases} \tag{38}$$

The full gradient descent algorithm for optimizing $L_D(\mathbf{w}, b, \{\mathbf{bb}^i\}_{i=1}^M)$ over $\mathbf{w}, b$ is summarized in Algorithm 1.

**Algorithm 1** Stochastic gradient descent for $[\mathbf{w}, b]$ optimization

pick a random example $i$

use $(\mathbf{bb}^i)^*$ computed in the previous step to calculate $f(\mathbf{w}, b; \mathbf{x}^i, (\mathbf{bb}^i)^*)$

**if** $y_i f(\mathbf{w}, b; \mathbf{x}^i, (\mathbf{bb}^i)^*) \geq 1$ **then**
  $\mathbf{w} \leftarrow \mathbf{w} - \alpha \mathbf{w}$,
  $b \leftarrow b - \alpha b$
**else**
  $\mathbf{w} \leftarrow \mathbf{w} - \alpha(\mathbf{w} - C y^i \mathbf{R}(\mathbf{x}^i, (\mathbf{bb}^i)^*))$,
  $b \leftarrow b - \alpha(b - C y^i)$
**end if**
where $\alpha$ is learning rate.

The learning rate $\alpha$ is set to $0.05/it$, where $it$ is the iteration index.

**Step 3. Optimization with respect to bb**

In this step we optimize $L_D(\mathbf{w}, b, \{\mathbf{bb}^i\}_{i=1}^M)$ with respect to $\{\mathbf{bb}^i\}_{i=1}^M$ by doing inference for every bounding box $\mathbf{bb}^i$ independently in the following way:

$$(\mathbf{bb}^i)^* = \underset{\mathbf{bb}^i}{\operatorname{argmin}} \left[ \max(0, 1 - y^i f(\mathbf{w}, b; \mathbf{x}^i, \mathbf{bb}^i)) - \lambda M(\mathbf{x}^i, \mathbf{bb}^i) \right] \tag{39}$$

$$= \underset{\mathbf{bb}^i}{\operatorname{argmax}} (y^i f(\mathbf{w}, b; \mathbf{x}^i, \mathbf{bb}^i) + \lambda M(\mathbf{x}^i, \mathbf{bb}^i)) \tag{40}$$

Here we can see how the search for a bounding box balances between good feature response $f(\mathbf{w}, b; \mathbf{x}^i, \mathbf{bb}^i)$ across the bounding box area and a high saliency concentration $M(\mathbf{x}^i, \mathbf{bb}^i)$ of the same bounding box. Further, by applying $f(\mathbf{w}, b; \mathbf{x}^i, \mathbf{bb}^i) = \mathbf{w}^T \mathbf{R}(\mathbf{x}^i, \mathbf{bb}^i) + b$, the inference can be written as:

$$\underset{\mathbf{bb}^i}{\operatorname{argmax}} (y^i (\mathbf{w}^T \mathbf{R}(\mathbf{x}^i, \mathbf{bb}^i) + b) + \lambda M(\mathbf{x}^i, \mathbf{bb}^i)). \tag{41}$$

Since $\mathbf{R}(\mathbf{x}^i, \mathbf{bb}^i)$ is concatenation of features per frame, we get:

$$\underset{\mathbf{bb}^i}{\operatorname{argmax}} \left[ y^i (\sum_{t=1}^{nt} w_t^T \mathbf{r}(\mathbf{x}_t^i, bb_t^i) + b) + \lambda \sum_{t=1}^{nt} m(\mathbf{x}_t^i, bb_t^i) \right] \tag{42}$$

$$= \underset{\mathbf{bb}^i}{\operatorname{argmax}} \Big\{ \sum_{t=1}^{nt} \left[ y^i w_t^T \mathbf{r}(\mathbf{x}_t^i, bb_t^i) + \lambda m(\mathbf{x}_t^i, bb_t^i) \right] + y^i b \Big\}. \tag{43}$$

We can ignore $y^i b$:

$$= \sum_{t=1}^{nt} \underset{bb_t^i}{\operatorname{argmax}} \left[ y^i w_t^T \mathbf{r}(\mathbf{x}_t^i, bb_t^i) + \lambda m(\mathbf{x}_t^i, bb_t^i) \right], \tag{44}$$

and, therefore, the inference of an optimal bounding box positions across the whole videos comes down to inference of the optimal bounding box position per frame $t$:

$$(bb_t^i)^* = \underset{bb_t^i}{\operatorname{argmax}} \left[ y^i w_t^T \mathbf{r}(\mathbf{x}_t^i, bb_t^i) + \lambda m(\mathbf{x}_t^i, bb_t^i) \right]. \tag{45}$$

**Step 4. Algorithm iteration**

After step 3 the algorithm iterates between step 2 and step 3 until $it$ reaches the maximum number of iterations.

As our bounding box search space is constrained to fixed size bounding boxes, the time complexity of one iteration of our algorithm is $O(W \times H \times nt)$ (for a single training example), where $W$ is width, $H$ is height and $nt$ is the number of frames of a video. The complexity of our method is the same as in [74]. In practice we sample a fixed number of points across x-axis, y-axis and frames as described in 4.2.1. We sample 20 points across the x-axis, 10 points across the y-axis and 5 frames across the video. Therefore, in our case $W = 20$, $H = 10$ and $nt = 5$.

**Classification**

Once learning is performed, we end up with $C$ binary classifiers that are trained in *one-vs.-all* manner. Each of those classifiers parametrized by $(w_c, b_c)$ can be used in order to determine whether a video described by $\mathbf{x}^i$ depicts the action $c$ by solving the following optimization problem (for clarity, we omit the index $i$):

$$y_c^*, \mathbf{bb}_c^* = \underset{y \in \{+1,-1\}, \mathbf{bb}}{\operatorname{argmax}} [y f(\mathbf{w}_c, b_c, \mathbf{bb}; \mathbf{x}) + \lambda M(\mathbf{x}, \mathbf{bb})]. \tag{46}$$

In order to solve the multiclass classification problem, we find the label $c^*$ that gives the maximum response $f(\mathbf{w}_c, b_c; \mathbf{x}, \mathbf{bb}_c^*)$ for the video in question. That is, we solve:

$$c^* = \underset{c \in \{1,\ ...,\ C\}}{\operatorname{argmax}} f(\mathbf{w}_c, b_c; \mathbf{x}, \mathbf{bb}_c^*), \tag{47}$$

where $\mathbf{bb}_c^*$ is given by Eq. (46).

Finally, let us note that when the bounding boxes are fixed, the classification decisions are not influenced by the saliency costs. That is, the binary classification in Eq. (46) becomes a standard SVM classification, that is:

$$y_c^* = \underset{y \in \{+1,-1\}}{\operatorname{argmax}} [y f(\mathbf{w}_c, b_c; \mathbf{x}, \mathbf{bb}_c^*)], \tag{48}$$

or

$$y_c^* = \operatorname{sgn} f(\mathbf{w}_c, b_c; \mathbf{x}, \mathbf{bb}_c^*). \tag{49}$$

### 4.2.2 Comparison with latent Support Vector Machine[1]

In this section we will show the relation of our model to the latent SVM proposed in [1]. If we set $\lambda = 0$, the cost function of our model takes the following form:

$$\min_{\mathbf{w},b,\{\mathbf{bb}^i\}_{i=1}^M} \frac{1}{2}(\mathbf{w}^T\mathbf{w} + b^2) + \sum_{i=1}^{M} \big[\max(0, 1 - y^i f(\mathbf{w}, b; \mathbf{x}^i, \mathbf{bb}^i))\big], \tag{50}$$

while the one of latent SVM[1] is defined as follows:

$$\min_{\mathbf{w},b} \frac{1}{2}(\mathbf{w}^T\mathbf{w} + b^2) + \sum_{i=1}^{M} \big[\max(0, 1 - y^i \max_{\mathbf{bb}^i} f(\mathbf{w}, b; \mathbf{x}^i, \mathbf{bb}^i))\big]. \tag{51}$$

When searching for $(y^i)^*$ and $(\mathbf{bb}^i)^*$ our model searches over all possible combinations of $(y^i)^*$ and $(\mathbf{bb}^i)^*$ (analogous to (Eq. 46) when $\lambda = 0$; for clarity, in the rest of the section we omit index $i$):

$$y^*, \mathbf{bb}^* = \operatorname*{argmax}_{y\in\{+1,-1\},\mathbf{bb}} y f(\mathbf{w}, b, \mathbf{bb}; \mathbf{x}). \tag{52}$$

In contrast to this, in [1] the search for $y^*$ and $\mathbf{bb}^*$ is performed in two separate steps:

$$\mathbf{bb}^* = \operatorname*{argmax}_{\mathbf{bb}} f(\mathbf{w}, b, \mathbf{bb}; \mathbf{x}), \tag{53}$$

and

$$y^* = \operatorname*{argmax}_{y\in\{+1,-1\}} y f(\mathbf{w}, b, \mathbf{bb}^*; \mathbf{x}). \tag{54}$$

The formulation presented in [1] is equivalent to one of the two multiple instance

SVM formulations presented in [147]. In this work two modifications of SVMs that deal with multiple instance learning (MIL) problems are proposed. Generally, in a MIL scheme class labels are associated with sets of patterns, i.e. bags, instead of individual patterns. In that way pattern labels are only indirectly accessible through labels attached to bags. The first formulation presented in [147] is pattern centered and the second is bag centered. This means that in the first approach the goal is to maximize the usual pattern margin jointly over hidden label variables and a discriminative function. The second approach generalizes the notion of a margin to bags and the goal is to maximize the bag margin directly. Therefore, in the pattern centered formulation the margin of every pattern in a positive bag matters and in the bag centered formulation only one pattern per positive bag matters, i.e. one pattern will determine the margin of the bag.

Both in our formulation and in the formulation of [1, 147] one bag contains multiple instances (patterns) from a single video. A bag of a video $i$ consist of all possible bounding boxes $\mathbf{bb}^i$ extracted across a single video, i.e. one pattern from a video bag is $\mathbf{bb}^i$. When using the formulation of [1, 147] we are looking for one bounding box which generates the best feature response and determines the class of the whole video bag, i.e. we search for the strongest bounding box area response first and classify it afterwards. In contrast to this, we associate an additional saliency cost with all patterns, i.e. bounding boxes, in a bag and search for one bounding box which will have the best combination of feature response and saliency cost. This bounding box is then the pattern which determines the margin of the whole video bag.

We found that this is the main drawback of the model of [1, 147]: it is not possible to incorporate the saliency cost under the $\max_{\mathbf{bb}}$ term as it would add negative saliency in the cost for negative examples. On the other hand, this model searches for the strongest bounding box area response first and classifies it afterwards. By doing so, the models avoids choosing the strong negative response and classifying it as negative, as it can happen in our model. In our model we are trying to avoid this by adding saliency cost. At the end of section 4.2.3 we will report how the experimental results obtained with each of those models compare.

### 4.2.3 Experimental results

In this section we present the action recognition results obtained with the method presented in 4.2.1. We validate this method on two publicly available sports action datasets. First, we use the UCF sports dataset which we have used in 4.1.2 and for which there are recorded fixations available. Second, we use the Olympic sports datasets[133], for which there are no recorded gaze fixations available and therefore it was not possible to validate our learned features and saliency prediction in the MV framework. However, it is possible and we will show the usefulness of saliency prediction in the SVM framework on this dataset. We use the suggested split for training and testing available on the dataset webpage.

When predicting saliency on Olympic sports dataset we use 3D CNN for saliency prediction that is trained on UCF sports dataset. As features we use the one from [148], which are also deeply learned using a CNN. However, they are learned on ILSVRC-2012 dataset, which contains only static 2D images, so they do not capture motion. Feature representation is built in the same way as in the experiments on the UCF sports dataset. The only minor difference is that during bounding box search $W$ and $H$ are both set to 7. This is due to the architectural properties of the network used for feature extraction (for more details see [148]).

As a baseline, in Table 8 we report the results obtained without optimizing with respect to the bounding box, i.e. when the bounding box at each frame is the whole frame. Those results are significantly lower than the ones obtained when using bounding boxes, except when $\lambda = 0$. This illustrates the importance of both searching for discriminative areas in the video and the importance of adding the saliency cost.

As the videos are unsegmented, introducing the bounding boxes can reduce the amount of background noise in the feature representation that is fed into the SVM classifier. However, when $\lambda = 0$ the bounding box choice is not regularized by the saliency cost. This can lead to choosing a bounding box that contains background without the relevant action parts. The classifier is then driven only by the irrelevant background features and that makes misclassification rate higher than when using the whole frames.

Furthermore, we would like to illustrate the importance of adding the saliency

| Method | UCF sports | Olympic sports |
|---|---|---|
| Lan *et al.* [104] | 73.1 | - |
| Shapovalova *et al.* [105] | 75.3 | - |
| Raptis *et al.* [85] | 79.4 | - |
| Jain *et al.* [149] | 80.24 | - |
| Shapovalova *et al.* [74] (1 region) | 77.98 | - |
| Shapovalova *et al.* [74] (2 regions) | 82.14 | - |
| Ma *et al.* [150] | 81.7 | - |
| Ma *et al.* [151] | 89.4 | - |
| Lu *et al.* [152] | 93.6 | - |
| Jain *et al.* [153] | 74.4 | - |
| Gkioxari *et al.* [154] | 75.8 | - |
| Lan *et al.* [155] | 83.6 | - |
| Niebles *et al.* [133] | - | 72.1 |
| Laptev *et al.* [156] | - | 62.0 |
| Ni *et al.* [86] | - | 92.3 |
| Peng *et al.* [84] | - | 93.8 |
| Ours (no bounding box) | 67.62 | 60.45 |
| Ours ($\lambda = +\infty$) | 79.05 | 67.16 |
| Ours ($\lambda = 0.0$) | 66.31 | 59.7 |
| Ours ($\lambda = 5.0$) | 83.57 | 64.18 |
| Latent SVM | 68.57 | 61.19 |

Table 8: Comparison of the results we obtain with our SVM approach to the state-of-the-art. The measure for UCF sports dataset is mean per class classification accuracy. The measure for Olympic sports dataset is mean average precision.

cost term in our SVM framework by varying the value of the parameter $\lambda$ in the set of values: $\{0.0, 2.5.5.0, 7.5, 10.0, 15.0, 17.5, 20.0\}$. The results obtained on the test set of the UCF sports dataset are presented in Fig. 29. We observe that increasing $\lambda$ significantly improves the results and that the peak is reached at $\lambda = 2.5$. For $\lambda$>17.5 the performance drops and then, as $\lambda \to +\infty$ it saturates. The highest performance on the test set was obtained for $\lambda = 2.5$ (85.24%), however with cross validation we obtained $\lambda = 5.0$, so in Table 8 we report the results obtained with that value of $\lambda$ (83.57%). The value of the SVM parameter $C$ was also obtained by cross validation; $C = 0.1$. Parameters $\lambda$ and $C$ were obtained by cross validation simultaneously and the cross validation was 2-folded.

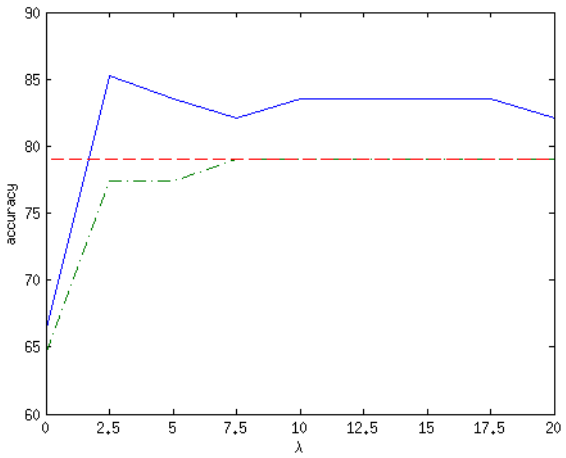



Figure 29: The effect of different $\lambda$ values on UCF sports test set. Dash-dot green line represents the results obtained before the optimization, i.e. using the initial weights and different values of $\lambda$, full blue line represents the results after the optimization, and red dashed line represents the result obtained using the initial weights and the initial bounding boxes.

In the same figure we illustrate the importance of optimization procedure, i.e. incorporating the bounding box search in the inference during both training and testing rather than just during testing, by the difference between the dash-dot green line and full blue line: the results after the optimization (full blue line) are consistently better then the results before the optimization (dash-dot green line) even though the initial weights are obtained using saliency (see Eq. (34)). Note that when using no saliency for the bounding box inference, i.e. for $\lambda = 0.0$, the result, that is 66.31%, is worse than the result obtained with majority voting using saliency prediction, that is 74.17% (see Table 6). As the initial weights were obtained using saliency information, this also illustrates the importance of saliency being incorporated in the optimization procedure.

For the Olympic sports dataset we perform cross validation only over $C$ parameter and use $\lambda = 5.0$ obtained for UCF sports dataset. The results we obtain on this dataset (see Table 8) also show that it is beneficial to use saliency, even if it is learned on a different dataset. We have seen that on the UCF sports best results are achieved for smaller $\lambda$. Therefore, it is interesting to note that on the Olympic sports dataset

| | d | g | k | l | ri | ru | s | sB | sSA | w |
|---|---|---|---|---|---|---|---|---|---|---|
| diving | 100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| golf | 0 | 83.3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 16.7 |
| kicking | 16.7 | 0 | 66.7 | 0 | 0 | 0 | 0 | 0 | 0 | 16.7 |
| lifting | 0 | 0 | 0 | 100 | 0 | 0 | 0 | 0 | 0 | 0 |
| riding | 0 | 0 | 0 | 0 | 100 | 0 | 0 | 0 | 0 | 0 |
| running | 0 | 0 | 0 | 0 | 0 | 75 | 0 | 0 | 0 | 0 |
| skateboarding | 0 | 0 | 0 | 0 | 0 | 0 | 25 | 0 | 0 | 75 |
| swing-bench | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 0 | 0 |
| swing-SA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 | 0 |
| walking | 0 | 14.3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 85.7 |

Table 9: Confusion matrix obtained for $\lambda = 5.0$ on the UCF sports dataset

better results are achieved when $\lambda = +\infty$, even though the saliency is learned on the UCF sports. This is probably due to the fact that in the experiments on UCF sports we use features trained on this dataset, and in the experiments on the Olympic sports we use features that are not trained on the Olympic sports dataset.However, it also implies that our learned saliency is general enough to improve the result on a different dataset comparing to the method that does not use saliency.

In the same table, that is Table 8, we also compare our results to the state-of-the-art. The work that is closest to ours is presented in [74]. In their work the recored human gaze fixations are incorporated, in the training phase only, in the form of a structural loss of a structured output latent SVM formulation that is used for action classification. That makes the gaze inference necessary during the optimization. By contrast, in our method the saliency prediction depends only on the output of the pretrained 3D CNN, i.e. there is no top-down inference. The works of [104, 105] also use latent SVM, however no saliency data has been used in either training or testing. [149, 85, 150, 151, 152, 154, 155, 153] also do not use saliency data. Another major difference in comparison to [74] is that as a feature representation they use BoW per frame. By contrast, we use feature concatenation, which seems to be a better representation, as inside the discriminative area of a bounding box the spatial relations should not be disregarded. However, our feature representation has obvious disadvantages which will be discussed later on. Furthermore, [74] reports the results obtained when inferring one and two discriminative regions to illustrate the

importance of adding flexibility in the choice of discriminative regions. We can see that even when using two discriminative regions (for which they obtain best results), their results were worse than ours even though we use only a single bounding box. In the confusion matrix presented in Table 9 we can see that the action kicking which contains additional object of interest (the ball) has the lowest accuracy.

We can see that our results on the UCF sports dataset are comparable to the ones reported in the recent work of [155] and outperformed by the ones reported in other two recent works: [151] and [152]. [151] and [155] share similar pipelines. In the first step they both, using different methods, generate an initial hierarchy of spatiotemporal segments which contain action elements such as actors, body parts and objects. In the second step they learn spatial, temporal and hierarchical relations upon the extracted spatiotemporal segments, [151] using tree based approach, and [155] using a discriminative clustering method. However, [155] requires supervision in terms of segmented data: when generating spatiotemporal segments it uses object proposal models trained on a dataset of segmented objects[157]. The highest results on the UCF sports dataset are reported in the work of [152]. The main focus of this work is generating a segmentation mask based on human motion saliency cues. The reported results (93.6%) are achieved by extracting dense trajectory features across the area segmented by the proposed method and using a $\chi^2$ SVM classifier on top of them. It is interesting to note that the results obtained by extracting dense trajectory features across the whole videos are 83.00%[152] and the results obtained by extracting dense trajectory features across the ground truth bounding boxes are 89.00%[152]. The findings that using bounding box or segmentation mask can generally improve classification accuracy over the whole video representation and that whole-human segmentation is better than a bounding box are consistent with evaluations from [158]. They are also consistent with our finding that using bounding boxes is a better than using whole frames for feature extraction. Note that the results in all three cases are high, even in the case of extracting features across the whole video. This implies the strength of trajectory based features. In addition to that, their segmentation method requires supervision in terms of bounding box annotations of humans. Namely, the human motion saliency is based on the output of DPM[1] which requires bounding box annotations of humans during training. Our method does not require bounding box annotations during training.

The state-of-the-art on Olympic sports dataset are obtained using trajectory based features and Fisher Vectors as a higher level video representation [84, 86]. [3] As we mention in the beginning of the related work section, those are the methods that in general currently hold or are close to the state-of-the-art on the action recognition datasets.

Overall, we observe that trajectory based features and modeling more complex spatiotemporal structures between feature extraction and classification stages are important for obtaining high action classification results. According to that, there could be a couple of possible directions for the future work in the scope of our SVM framework. First would be to look into ways how to incorporate trajectory based and FV based representations in our framework. We could also consider incorporating other types of feature hierarchies upon the local action features and modifying the SVM to act simultaneously on multiple levels of hierarchies, possibly even to learn how to model the hierarchies. Second, as it is found that using segmentation mask is better than using bounding boxes (even ground truth boundig boxes), it would be good to incorporate more flexibility in the bounding box search space of our framework, especially in the number and the size of the bounding boxes. This indeed is the obvious limitation of our approach: the choice of a fixed size bounding box makes our representation sensitive to scale changes (see Fig. 27 - the videos presented in this figure are not misclassified only in the majority voting scheme, but in our SVM scheme too). Another downside of our approach is using fixed feature concatenation without any kind of pooling. This makes our feature representation sensitive to translation changes. That is a problem especially for periodic actions, such as skateboarding and running - in Table 9 we can see that accuracies obtained for those actions are lower. Using FV representation or other more sophisticated spatiotemporal structures could solve this problem.

**Comparing with [1]**

In section 4.2.2 we showed how our model in a special case when $\lambda = 0$ compares to the one of [1]. Here, in Table 8 we report how their results compare. As we are interested in comparing only the performance of the latent SVM model [1] to ours, the features that are used are the ones that we used in all our experiments, i.e. the

[3] Note that the results of the approach of [84] we report are reported in [159], not in [84].

ones obtained by 3D CNN. That is, we do not implement the deformable parts model on top of which latent SVM is built, as presented in [1].

We can see that the model of [1] achieves slightly better results than our model in case of $\lambda = 0$. This was expected as our model can pick up on the noise of strong negative bounding box responses, while [1] searches for the strong bounding box area response first and classifies it afterwards. However, with added saliency cost, our model outperforms it by a large margin. Hence, it seems that adding saliency cost acts as a much better regularization scheme for not picking the irrelevant background clutter.

### 4.2.4 Conclusions

We have developed an SVM framework which incorporates the saliency cost from our learned saliency prediction and representation built from learned action features. In this framework we have shown the following.

First, by comparing the results obtained when using features extracted across bounding boxes fixed at the most salient areas (predicted by our 3D CNN) with the results obtained when using features extracted across the whole videos we have shown the importance of using bounding boxes and saliency.

Second, by comparing the results obtained with the optimization where bounding boxes are considered as latent variables with the results obtained with the optimization where bounding boxes are fixed at the most salient areas we have shown the efficacy of the modeling that jointly optimizes with respect to the standard SVM weights and the location of bounding boxes that capture the action.

Third, by comparing the results of our SVM obtained with incorporated saliency cost with the results obtained without incorporating the saliency cost, we have shown the importance of using saliency in our SVM framework.

Fourth, by comparing our results with the results of [1] we have shown that adding saliency cost acts as a much better regularization scheme for not picking the irrelevant background clutter than only searching for the strong bounding box area response, as in [1].

However, the main disadvantage of our work is building a representation by concatenation. Building a representation that consist of features concatenated across

fixed number of frames and points across x and y axis of a frame may not be the best representation for some type of actions. As mentioned in section 4.2.3 our current representation is not robust to larger translation and scale variations. Also, some actions may require a representation that consists of multiple bounding boxes. In addition to that, the learned action features themselves are not perfect, as mentioned in section 4.1.2. As we mention when analyzing state-of-the-art results, few things that may help to improve the action recognition results are: using trajectories and FV representations or building other more sophisticated spatiotemporal hierarchical structures, and using more fine grained segmentation than a single fixed size bounding box.

# 5 Conclusions

In this thesis we have investigated how visual saliency can be learned and used in the action recognition framework to alleviate action localization and classification. In chapter 3 we have focused on learning visual saliency using different learning criteria on different levels of the saliency prediction architecture. In chapter 4 we have focused on learning action features using gaze information and on the development of an SVM based action recognition framework that uses learned action features and learned saliency.

In chapter 3 we have focused on the saliency prediction.

First, in contrast to the traditional approaches which use manually selected features for saliency prediction, we employed an unsupervised learning algorithm, that is topographic ICA, in order to learn the low level features for saliency prediction. Those features have shown superior performance compared to the combination of manually selected low, mid and high level features. Second, we proposed to learn the mid layer tICA weights, which are originally kept fixed, and the SVM weights jointly. As an optimization criterion we have used the SVM supervision criterion. This means that both the SVM weights and the mid layer tICA weights were optimized to correctly classify the fixations and non fixations. For the purpose of such joint learning we have developed an iterative block coordinate descent optimization procedure. We have shown that such joint learning yields superior saliency prediction results over the learning in which tICA feature learning and SVM learning are separated. Third, we have used a 2D CNN in which all layers are optimized jointly to correctly classify the fixations and non fixations. We have shown that such feature learning that uses only the supervision criterion on all layers yields the best performance in the saliency prediction task. Fourth, we have extended 2D convolution to 3D and compared the performance of 2D and 3D CNNs in the task of spatiotemporal saliency prediction. We have shown that 3D CNN yields better saliency prediction results.

In chapter 4 we have focused on the action recognition problem and how to use the saliency to alleviate this problem.

First, as opposed to the commonly used manually designed local action features,

such as HOG/HOF, we have learned 3D action features using a fully supervised 3D CNN. The network was optimized to discriminate cuboids according to their action class. We have shown that features learned in such way compare to the manually designed action features.

Second, we have proposed an SVM-based recognition framework for joint recognition and localization, in which the bounding box of the action is considered as a latent variable. We have developed an optimization procedure which attempts to both minimize the classification cost and maximize the saliency within the bounding box. We have shown that the action recognition results obtained with the optimization where saliency within the bounding box is maximized outperform the action recognition results obtained when saliency within the bounding box is not maximized, i.e. when only classification cost is minimized. Furthermore, this approach has shown better results than the latent SVM proposed in [1].

To summarize, the work presented in this thesis shows that:

1. learned saliency features yield better results than manually selected saliency features (S1),

2. incorporating the supervision at more layers of a saliency learning architecture improves the saliency prediction results (S2, S3),

3. learned action features yield better results than manually selected action features (A1),

4. using learned saliency improves action recognition results in a couple of action recognition frameworks (A1, A2).

The marks in the parenthesis of the each point show to which contribution (listed in 1.2) the point is related. Also, we again refer to Fig. 1 where it can be seen which part of the work supports which point.

## 5.1 Future work

First, we describe the future work regarding saliency prediction.

In our experiments regarding 2D saliency prediction we have used shallow architectures: the topographic ICA architecture is a two layer network, and the CNN architecture is a five layer network in which two layers are fixed pooling layers. Those architectures cannot yield very high level features. It would be interesting to see if combining our learned features with manually selected high level features, such as face detection, would improve the saliency prediction results. The next step could be building deeper networks and see two things. First, how much adding layers can improve the saliency prediction and second, if there is a certain depth at which either adding layers or combining learned features with manually selected features does not improve the results.

In the problem of 2D saliency prediction we were dealing with natural images for which the gaze fixations were recorded in a free viewing manner, i.e. the people observing the images were not given a specific task. The gaze fixations for the UCF sports dataset were recorded in the same way. However, in the UCF sports dataset there are different (action) classes of videos and that may influence the viewers, i.e. the action classes may account for the top-down inference. It would be interesting to investigate further how to design and implement different architectural changes that would take into consideration action dependent top-down inference.

Second, we describe the future work regarding action recognition and combining saliency prediction and action recognition.

In our SVM based action recognition framework we are building a representation that consist of features concatenated across fixed number of frames and points across a frame. As mentioned in section 4.2.3 this representation has few disadvantages. First, it is not robust to larger translation and scale variations. Second, some actions may require a representation that consists of multiple bounding boxes. Third, the learned action features themselves are not perfect. According to those observations, there are few possible research directions. First is enhancing the training of the local action features by using deeper architectures or by using different cuboid sampling strategy (we mention in 4.1.2 that some fixation points might capture irrelevant movements

which may corrupt the learning - therefore including all fixation points in the training may not be the best strategy). Also, using more data which may or may not contain recorded gaze fixations, could be beneficial - one of the challenges could be how to use the data which does not contain the recorded gaze fixations. Second is extending our SVM framework to incorporate the search over multiple bounding boxes of different sizes. Third is using different representation upon the learned features. As we mention in 4.2.3 the representations that yield the state of the art are based on trajectories and Fisher Vectors. Therefore, one of the possible directions would be to investigate how to incorporate such representations in our framework.

It would be also interesting to see how the saliency learning and action recognition can be unified in a way that the learning of the action class and/or action features is incorporated in the learning of the saliency and vise versa. One way to achieve this could be to develop a learning procedure that iterates between the saliency learning in which the predicted action class/features is used and the action learning in which predicted saliency is used. For example, we could iterate between our SVM learning and saliency learning. Within this procedure it should be learned how to use predicted saliency for action recognition and predicted action class for saliency prediction.

# References

[1] P. Felzenschwalb, R. Girshick, D. McAllester, and D. Ramannan, "Object detection with discriminatively trained part based models", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2009.

[2] Q. Zhao and C. Koch, "Learning visual saliency by combining feature maps in a nonlinear manner using adaboost", *Journal of Vision*, vol. 12, no. 6, pp. 1–15, 2012.

[3] W. Kienzle, B. Scholkopf, F. Wichmann, and M. Franz, "How to find interesting locations in video: a spatiotemporal interest point detector learned from human eye movements", in *Proceedings of DAGM*, 2007.

[4] S. Haykin, *Neural Networks and Machine Learning*, Pearson, Third edition, ISBN: 978-0-13-129376-2, 2008.

[5] H. Lee, C. Ekanadham, and A. Ng, "Sparse deep belief net model for visual area v2", in *Proceedings of Neural Information Processing Systems Conference*, 2007.

[6] H. Lee, R. Grosse, , R. Ranganath, and A. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hiererchical representations", in *Proceedings of International Conference on Machine Learning*, 2009.

[7] Q. Le, W. Zou, S. Yeung, and A. Ng, "Learning hierarchihal invariant spatio-temporal features for action recognition with independent subspace analysis", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2011.

[8] A. Hyvarinen, Y. Hurri, and P. Hoyer, *Natural Image Statistics*, Springer, ISBN: 978-1-84882-491-1, 2009.

[9] "theano - deep learning tutorial", `http://deeplearning.net/tutorial`.

[10] A. Jaimes, J. Pelz, T. Grabowski, J. Babcock, and S. Chang, "Using human observers' eye movements in automatic image classifiers", in *Proceedings of SPIE Human Vision and Electronic Imaging VI*, 2001.

[11] A. Klami, C. Saunders, T.E. Campos, and S. Kaski, "Can relevance of images be inferred from eye movements?", in *Proceedings of International Conference on Multimedia Information Retrieval*, 2008.

[12] Y. Zhang, H. Fu, Z. Liang, Z. Chi, and D. Feng, "Eye movement as an interaction mechanism for relevance feedback in a content-based image retrieval system", in *Proceedings of the Eye Tracking Research and Application Symposium*, 2010.

[13] S. Vrochidis, I. Patras, and I. Kompatsiaris, "Exploiting gaze movements for automatic video annotation", in *Proceedings of International Workshop on Image Analysis for Multimedia Interactive Services*, 2012.

[14] M. Sadeghi, G. Tien, G. Hamarneh, and M.S. Atkins, "Hands-free interactive image segmentation using eyegaze", in *Proceedings of SPIE 7260, Medical Imaging 2009: Computer-Aided Diagnosis*, 2009.

[15] A. Fathi, Yin. Li, and James M. Rehg, "Learning to recognize daily actions using gaze", in *Proceedings of European Conference on Computer Vision*, 2012.

[16] K. Yun, Y. Peng, D. Samaras, G. Zelinksy, and T. Berg, "Studying relationships between human gaze, description, and computer vision", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2013.

[17] K. Yun, Y. Peng, D. Samaras, G. Zelinksy, and T. Berg, "Exploring the role of gaze behavior and object detection in scene understanding", *Frontiers in Psychology*, vol. 4, pp. 1–14, 2013.

[18] E. Vig, M. Dorr, and D. Cox, "Space-variant descriptor sampling for action recognition based on saliency and eye movements", in *Proceedings of European Conference on Computer Vision*, 2012.

[19] G. Ge, K. Yun, D. Samaras, and G. Zelinksy, "Action classification in still images using human eye movements", in *The 2nd Vision Meets Cognition Workshop at Conference on Computer Vision and Pattern Recognition*, 2015.

[20] L. Itti, "Visual salience", vol. 2, no. 9, pp. 3327, 2007.

[21] L. Itti and C. Koch, "Computational modeling of visual attention", *Nature Reviews Neuroscience*, vol. 2, pp. 194–203, 2001.

[22] J. Wolfe and T. Horowitz, "What attributes guide the deployment of visual attention and how do they do it?", *Nature Reviews Neuroscience*, vol. 5, pp. 1–7, 2004.

[23] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis", *IEEE Transactions on Pattern Analysis and Machine Learning*, pp. 1254–1259, 1998.

[24] D. Gao, V. Mahadevan, and N. Vasconcelos, "The discriminant center-surround hypothesis for bottom-up saliency", in *Proceedings of Neural Information Processing Systems Conference*, 2007.

[25] N. Bruce and J. Tsotsos, "Saliency based on information maximization", in *Proceedings of Neural Information Processing Systems Conference*, 2005.

[26] X. Sun, H. Yao, and R. Ji, "Measuring visual saliency by site entropy rate", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2010.

[27] S. Assari, A. Zamir, and M. Shah, "Video classification using semantic concept co-occurrences", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2014.

[28] G. Perazzi, P. Krahenbuhl, Y. Pritch, and A. Hornung, "Saliency filters: Contrast based filtering for salient region detection", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2012.

[29] E. Eleonora Vig, M. Dorr, and D. Cox, "Large-scale optimization of hierarchical features for saliency prediction in natural images", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2014.

[30] M. Kmmerer, L. Theis, and M. Bethge, "Deep gaze i: Boosting saliency prediction with feature maps trained on imagenet", in *arXiv:1411.1045*, 2014.

[31] C. Shen, M. Song, and Q. Zhao, "Learning high-level concepts by training a deep network on eye fixations", in *Deep Learning and Unsupervised Feature Learning Workshop, in conjunction with NIPS*, 2012.

[32] C. Shen and Q. Zhao, "Learning to predict eye fixations for semantic contents using multi-layer sparse network", *Neurocomputing*, vol. 138, pp. 61–68, 2014.

[33] W. Kienzle, F. A. Wichmann, B. Scholkopf, and F. O. Matthias, "A non-parametric approach to bottom-up visual saliency", in *Proceedings of Neural Information Processing Systems Conference*, 2007.

[34] W. Kienzle, F. A. Wichmann, B. Scholkopf, and F. O. Matthias, "Learning an interest operator from human eye movements", in *In Beyond Patches Workshop, International Conference on Computer Vision and Pattern Recognition*, 2006.

[35] W. Kienzle, F. O. Matthias, B. Scholkopf, and F. A. Wichmann, "Center-surround patterns emerge as optimal predictors for human saccade targets", *Journal of Vision*, vol. 9, no. 5, pp. 1–15, 2009.

[36] T. Judd, K. Ehinger, F. Durand, and A. Torralba, "Learning to predict where humans look", in *Proceedings of International Conference on Computer Vision*, 2009.

[37] Q. Zhao and C. Koch, "Learning a saliency map using fixated locations in natural scenes", *Journal of Vision*, vol. 11, no. 3, pp. 1–15, 2011.

[38] Q. Zhao and C. Koch, "Learning saliency based visual attention: A review", *Signal Processing*, vol. 93, no. 6, pp. 1401–1407, 2012.

[39] M. Cerf, J. Harel, W. Einhauser, and C. Koch, "Predicting human gaze using low-level saliency combined with face detection", in *Proceedings of Neural Processing Systems Conference*, 2007.

[40] A. Borji, D. Sihite, and L. Itti, "Quantitative analysis of human-model agreement in visual saliency modeling: A comparative study", *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 55–69, 2013.

[41] S. Nataraju, V. Balasubramanian, and S. Panchanatan, "Learning attention based saliency in videos from human eye movements", in *Workshop on Motion and Video Computing*, 2009.

[42] D. Rudoy, D. B. Goldman, E. Shechtman, and L. Zelnik-Manor, "Learning video saliency from human gaze using candidate selection", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2013.

[43] S. Mathe and C. Sminchisescu, "Dynamic eye movement datasets and learnt saliency models for visual action recognition", in *Proceedings of European Conference on Computer Vision*, 2012.

[44] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features", in *Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

[45] I. Laptev and T. Lindeberg, "Space-time interest points", in *Proceedings of International Conference on Computer Vision*, 2003.

[46] A. Hyvarinen, P. Hozer, and M. Inki, "Topographic ica as s model of v1 receptive fields", in *Proceedings of International Joint Conference on Neural Networks*, 2000.

[47] R. Mutihac and M. M. van Hulle, "Statistics of feature extraction by topographic independent component analysis from natural images", in *Proceedings of International Conference on Electronics Control and Signal Processing*, 2003.

[48] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, Elsevier, ISBN 13: 978-0-12-374370-1, 2009.

[49] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images", *Nature*, pp. 607–609, 1996.

[50] B. A. Olhausen, "Sparse coding of time-varying natural images", in *Proceedings of the International Conference on Independent Component Analysis and Blind Source Separation*, 2000.

[51] M. Jiang, M. Song, and Q. Zhao, "Leveraging human fixations in sparse coding: Learning a discriminative dictionary for saliency prediction", in *IEEE International Conference on Systems, Man, and Cybernetics*, 2013.

[52] K. Guo and H. Chen, "Learning sparse dictionaries for saliency detection", in *Signal and Information Processig Association Annual Summit and Conference*, 2012.

[53] A. Carbone and F. Pirri, "Learning saliency. an ica based model using bernoulli mixtures", in *Proceedings of Brain Inspired Cognitive Systems*, 2010.

[54] C. Kanan and G. Cottrell, "Robust classification of objects, faces, and flowers using natural image statistics", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2010.

[55] A. Saxe, P. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Ng, "On random weights and unsupervised feature learning", in *Proceedings of International Conference on Machine Learning*, 2011.

[56] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional neural netoworks", in *Proceedings of International Joint Conference on Neural Networks*, 2011.

[57] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets", *Neural computation*, vol. 18, pp. 1527–1554, 2006.

[58] D. Erhan, Y. Bengio, A. Courville, P. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?", *Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[59] H. Larochelle, Bengio. Y., J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks", *Journal of Machine Learning Research*, vol. 1, pp. 1–40, 2009.

[60] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning", in *Proceedings of International Conference on Machine Learning*, 2013.

[61] P. Lamblin and Y. Bengio, "Important gains from supervised fine-tuning of deep architectures on large labeled sets", in *Proceedings of Neural Information Processing Systems Conference*, 2010.

[62] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks", in *Proceedings of Neural Information Processing Systems Conference*, 2012.

[63] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection", in *Proceedings of Neural Information Processing Systems Conference*, 2013.

[64] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition", *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[65] Yoshua Bengio and Yann LeCun, "Convolutional networks for images, speech, and time-series", 1995.

[66] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch", *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[67] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng, "Self-taught learning: Transfer learning from unlabeled data", in *Proceedings of International Conference on Machine Learning*, 2007.

[68] J. Bruna and S. Mallat, "Invariant scattering convolution networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 1872–1886, 2013.

[69] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", in *Proceedings of the IEEE*, 1998.

[70] K. P. Murphy, *Machine Learning. A Probabilistic Approach.*, The MIT Press, ISBN: 978-0262018029, 2012.

[71] Y. Lin, S. Kong, D. Wang, and Y. Zhuang, "Saliency detection within a deep convolutional architecture", in *Proceedings of AAAI Workshop on Cognitive Computing for Augmented Human Intelligence*, 2014.

[72] R. Zhao, W. Ouyang, H. Li, and X. Wang, "Saliency detection by multi-context deep learning", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2015.

[73] S. Winkler and R. Subramanian, "Overview of eye tracking datasets", in *Proceedings of Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*, 2013.

[74] N. Shapovalova, M. Raptis, L. Sigal, and G. Mori, "Action is in the eye of the beholder: Eye-gaze driven model for spatio-temporal action localization", in *Proceedings of Neural Information Processing Systems Conference*, 2013.

[75] J.K. Aggarwal and M. S. Ryoo, "Human activity analysis: A review", *ACM Computing Surveys*, vol. 43, 2011.

[76] R. Poppe, "A survey on vision-based human action recognition", *Image Vision Computing*, vol. 28, pp. 976–990, 2010.

[77] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition", *Computer Vision and Image Understanding*, vol. 115, pp. 224–241, 2011.

[78] J. C. Niebles, H. Wang, and L. Fei-Fei, "Unsupervised learning of human action categories using spatial-temporal words", *Internation Journal of Computer Vision*, vol. 79, no. 3, pp. 299–318, 2008.

[79] H. Wang, Ullah M. M., A. Klaser, I. Laptev, and C. Schmid, "Evaluation of local spatio-temporal features for action recognition", in *Proceedings of British Machine Vision Conference*, 2009.

[80] H. Wang, A.Klaser, C.Schmid, and C. Liu, "Action recognition by dense trajectories", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2011.

[81] H. Wang, A.Klaser, C.Schmid, and C. Liu, "Action recognition with improved trajectories", in *Proceedings of International Conference on Computer Vision*, 2013.

[82] J. Sun, X. Wu, S. Yan, L. Cheong, T. Chua, and J. Li, "Hierarchical spatio-temporal context modeling for action recognition", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2009.

[83] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice", in *arXiv preprint arXiv:1405.4506*, 2014.

[84] X. Peng, C. Zou, Y. Qiao, and Q. Peng, "Action recognition with stacked fisher vectors", in *Proceedings of European Conference on Computer Vision*, 2014.

[85] M. Raptis, I. Kokkinos, and S. Soatto, "Discovering disriminative action parts from mid-level video representation", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2012.

[86] B. Ni, P. Moulin, X. Yang, and S. Yan, "Motion part regularization: Improving action recognition via trajectory group selection", in *Proceedings International Conference on Computer Vision and Pattern Recognition*, 2015.

[87] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors", in *Proceedings International Conference on Computer Vision and Pattern Recognition*, 2015.

[88] F. Perronnin, J. Sanchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification", in *Proceedings of European Conference on Computer Vision*, 2010.

[89] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods", in *Proceedings of British Machine Vision Conference*, 2011.

[90] D. Oneata, J. Verbeek, and C. Schmid, "Action and event recognition with fisher vectors on a compact feature set", in *Proceedings of International Conference on Computer Vision*, 2013.

[91] H. Wang, D. Oneata, J. Verbeek, and C. Schmid, "A robust and efficient video representation for action recognition", in *arXiv preprint arXiv:1504.05524*, 2015.

[92] A. Quattoni, S. Wang, L. Morency, M. Collins, and T. Darrell, "Hidden-state conditional random fields", *IEEE TPAMI*, 2007.

[93] C. Sminchisescu, A. Kanaujia, and D. Metaxas, "Conditional models for contextual human motion recognition", *CVIU*, 2006.

[94] Y. Song, L. Morency, and R. Davis, "Action recognition by hierarchical sequence summarization", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2013.

[95] Y. Wang and G. Mori, "Hidden part models for human action recognition: Probabilistic versus max margin", *IEEE TPAMI*, 2011.

[96] D. Q. Phung T. V. Duong, H. H. Bui and S. Venkatesh, "Activity recognition and abnormality detection with the switching hidden semi-markov model", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2005.

[97] S. Hongeng and R. Nevatia, "Large-scale event detection using semi-hidden markov models", in *Proceedings of International Conference on Computer Vision*, 2003.

[98] P. Natarajan and R. Nevatia, "Coupled hidden semi markov models for activity recognition", in *Proceedings of IEEE workshop on Motion and Video Computing (WMVC)*, 2007.

[99] K. Tang, L. Fei-Fei, and D. Koller, "Learning latent temporal structure for complex event detection", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2012.

[100] J. K. Aggarwal and M. S. Ryoo, "Recognition of composite human activities through context-free grammar based representation", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2006.

[101] H. Pirsiavash and D. Ramanan, "Parsing videos of actions with segmental grammars", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2014.

[102] H. Kuehne, A. Arslan, and T. Serre, "The language of actions: Recovering the syntax and semantics of goal-directed human activities", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2014.

[103] N. N. Vo and A. F. Bobick, "From stochastic grammar to bayes network: Probabilistic parsing of complex activity", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2014.

[104] T. Lan, Y. Wnag, and G. Mori, "Discriminative figure-centric models for joint action localization and recognition", in *Proceedings of International Conference on Computer Vision*, 2011.

[105] N. Shapovalova, A. Vahdat, K. Cannons, T. Lan, and G. Mori, "Similarity constrained latent support vector machine: An application to weakly supervised action classification", in *Proceedings of European Conference on Computer Vision*, 2012.

[106] A. Gilbert, J. Illingworth, and R. Bowden, "Action recognition using mined hierarchical compound features", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 883–897, May 2011.

[107] A. Kovashka and K. Grauman, "Learning a hierarchy of discriminative space-time neighborhood features for human action recognition", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2010.

[108] S. Christian, I. Laptev, and B. Caputo, "Recognizing human actions: A local svm approach", in *Proceedings of International Conference on Pattern Recognition*, 2004.

[109] M. Sapienza, F. Cuzzolin, and P. Torr, "Learning discriminative space-time actions from weakly labelled videos", in *Proceedings of British Machine Vision Conference*, 2012.

[110] J. Yuan, L. Zicheng, and Y. Wu, "Discriminative subvolume search for efficient action detection", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2009.

[111] S. Koelstra and I. Patras, "The fast-3d spatio-temporal interest region detector", in *Workshop on Image Analysis for Multimedia Interactive Services*, 2009.

[112] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2014.

[113] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition", in *Proceedings of Workshop on Human Behaviour Understanding*, 2011.

[114] L. Sun, K. Jia, T. Chan, Y. Fang, G. Wang, and S. Yan, "Dl-sfa: Deeply-learned slow feature analysis for action recognition", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2014.

[115] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features", in *Proceedings of European Conference on Computer Vision*, 2010.

[116] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks", in *Proceedings of International Conference on Computer Vision*, 2015.

[117] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos", in *Proceedings of Neural Information Processing Systems Conference*, 2014.

[118] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computing*, vol. 9, pp. 1735–1780, 1997.

[119] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sparse shift invariant representation of local 2d patterns and sequence learning for human action recognition", in *Proceedings of International Conference on Pattern Recognition*, 2012.

[120] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Spatio-temporal convolutional sparse auto-encoder for sequence classification", in *Proceedings of British Machine Vision Conference*, 2012.

[121] G. Cheron, I. Laptev, and C. Schmid, "P-cnn: Pose-based cnn features for action recognition", in *Proceedings of International Conference on Computer Vision*, 2015.

[122] K. Soomro, A. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild", in *abs/1212.0402*, 2012.

[123] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "Hmdb: A large video database for human motion recognition", in *Proceedings of International Conference on Computer Vision*, 2011.

[124] V. Veeriah, N. Zhuang, and G. Qi, "Differential recurrent neural networks for action recognition", in *Proceedings of International Conference on Computer Vision*, 2015.

[125] J. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification", in *Proceedings International Conference on Computer Vision and Pattern Recognition*, 2015.

[126] S. Sharma, R. Kiros, and R. Salakhutdinov, "Action recognition usig visual attention", in *Proceedings of International Conference on Learning Representations*, 2016.

[127] D. Tran and L. Torresani, "Exmoves: Classifier-based features for scalable action recognition", in *Proceedings of International Conference on Learning Representations*, 2014.

[128] D. Tran and L. Torresani, "Exmoves: Mid-level features for efficient action recognition and video analysis", *International Journal on Computer Vision (IJCV)*, 2016.

[129] A. Jain, A. Gupta, M. Rodriguez, and L. Davis, "Representing videos using mid-level discriminative patches", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2013.

[130] C. Liu, Y. Kong, X. Wu, and Y. Jia, "Action recognition with discriminative mid-level features", in *Proceedings of International Conference on Pattern Recognition*, 2012.

[131] G. Sharma, F. Jurie, and C. Schmid, "Discriminative spatial saliency for image classification", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2012.

[132] L Zhu, Y. Chen, A. Yuille, and W. Freeman, "Latent hierarchical structural learning for object detection", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2010.

[133] J.C. Niebles, C. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments of activity classification", in *Proceedings of European Conference on Computer Vision*, 2010.

[134] M. H. Nguyen, L. Torresani, F. de la Toree, and C. Rother, "Weakly supervised discriminative localization and classification: a joint learning process", in *Proceedings of International Conference on Computer Vision*, 2009.

[135] N. Sebe, Q. Tian, E. Loupias, M. S. Lew, and T. S. Huang, "Evaluation of salient point techniques", *Image and Vision Computing*, vol. 21, pp. 1087–1095, 2003.

[136] D. Gao and N. Vasconselos, "Integrated learning of saliency, complex features, and object detectors from cluttered scenes", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2005.

[137] M.D. Rodriguez, J. Ahmed, and M. Shah, "Action mach a spatio-temporal maximum average correlation height filter for action recognition", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2008.

[138] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, ISBN: 978-0387987804, 1999.

[139] V. Kumar, I. Kotsia, and I. Patras, "Max-margin non-negative matrix factorization", *Image and Vision Computing*, vol. 30, pp. 279–291, 2012.

[140] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets", in *Proceedings of Neural Information Processing Systems Conference*, 2014.

[141] R. Fan, K. Chang, C. Hsein, X. Wang, and C. Lin, "Liblinear: A library for large linear classification", *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[142] O. Chapelle, "Training a support vector machine in the primal", *Neural Computation*, vol. 19, no. 5, 2007.

[143] Y. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition", in *Proceedings of International Conference on Machine Learning*, 2010.

[144] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio, "Theano: new features and speed improvements", Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[145] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio, "Theano: a CPU and GPU math expression compiler", in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

[146] T. Mauthner, H. Possegger, G. Waltner, and H. Bischof, "Encoding based saliency detection for videos and images", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2015.

[147] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning", in *Proceedings of Neural Information Processing Systems Conference*, 2002.

[148] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", in *Proceedings of International Conference on Learning Representations*, 2015.

[149] M. Jain, J. Gemert, H. Jegou, P. Bouthemy, and C. Snoek, "Action localization with tubelets from motion", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2014.

[150] S. Ma, J. Zhang, N. Ikizler-Cinbis, and S. Sclaroff, "Action recognition and localization by hierarchical space-time segments", in *Proceedings of International Conference on Computer Vision*, 2013.

[151] S. Ma, L. Sigal, and S. Sclaroff, "Space-time tree ensemble for action recognition", in *Proceedings International Conference on Computer Vision and Pattern Recognition*, 2015.

[152] J. Lu, R. Xu, and J. Corso, "Human action segmentation with hierarchical supervoxel consistency", in *Proceedings International Conference on Computer Vision and Pattern Recognition*, 2015.

[153] M. Jain, J. Gemert, and C. Snoek, "What do 15,000 object categories tell us about classifying and localizing actions?", in *Proceedings International Conference on Computer Vision and Pattern Recognition*, 2015.

[154] G. Gkioxari and J. Malik, "Finding action tubes", in *Proceedings International Conference on Computer Vision and Pattern Recognition*, 2015.

[155] T. Lan, Y. Zhu, Z. Roshan, and S. Savarese, "Action recognition by hierarchical mid-level action elements", in *Proceedings of International Conference on Computer Vision*, 2015.

[156] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2008.

[157] I. Endres and D. Hoiem, "Category independent object proposals", in *Proceedings of European Conference on Computer Vision*, 2010.

[158] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. Black, "Towards understanding action recognition", in *Proceedings of International Conference on Computer Vision*, 2013.

[159] Z. Lan, M. Lin, X. Li, A. Hauptmann, and B. Raj, "Beyond gaussian pyramid: Multi-skip feature stacking for action recognition", in *Proceedings of International Conference on Computer Vision and Pattern Recognition*, 2015.

# 6 Appendix A: Topographic ICA optimization criterion

Here, we will show how the topographic ICA optimization criterion, that is Eq. (2), is derived. All details of the derivation can be found in [8].

A vectorized image or an image patch $I$ is modeled as a linear superposition of features $\mathbf{a}_i$ with corresponding random coefficients $s_i$:

$$I = \sum_{i=1}^{n_1} \mathbf{a}_i s_i = s_1 \begin{bmatrix} a_{1,1} \\ a_{2,1} \\ ... \\ a_{n,1} \end{bmatrix} + s_2 \begin{bmatrix} a_{1,2} \\ a_{2,2} \\ ... \\ a_{n,2} \end{bmatrix} + ... + s_{n_1} \begin{bmatrix} a_{1,n_1} \\ a_{2,n_1} \\ ... \\ a_{n,n_1} \end{bmatrix}, \tag{55}$$

where $n_1$ is the number of features and $n$ is the total number of image pixels. The $s_i$ are coefficients that change randomly for each patch. They can thus be considered as random variables. In contrast, the features $\mathbf{a}_i$ are the same for all patches.

In what follows we consider whitened image patches instead of raw image patches $I$, i.e. we consider $\mathbf{z}$ which is a whitened image patch $I$. In that case a single feature $s_i$ is computed as:

$$s_i = \mathbf{v}_i^T \mathbf{z} = \sum_{j=1}^{n} v_{ij} z_j, \tag{56}$$

where $\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2\ ...\ \mathbf{v}_{n_1}]^T$ is an inverse of matrix $\mathbf{A}$. We assume that the matrix $\mathbf{A}$ is invertible, although this assumption does not always hold. However, for whitened image patches it does hold (for more details see [8]).

The goal is to estimate features $\mathbf{V}$ by maximizing the likelihood of the observed data. Let us assume that we have observed a single image patch $\mathbf{z}$. The likelihood $L$ of the observed patch is:

$$L(\mathbf{v}_1, \mathbf{v}_2, \ ..., \ \mathbf{v}_{n_1}) = p(\mathbf{z}) = p(s_1, s_2, \ ..., \ s_{n_1}) = \prod_{i=1}^{n_1} p(s_i). \tag{57}$$

From Eq. (57) and Eq. (56) follows (for more details see [8]):

$$L(\mathbf{v}_1, \mathbf{v}_2, \ ..., \ \mathbf{v}_{n_1}) = |\det(\mathbf{V})| \prod_{i=1}^{n_1} p(\mathbf{v}_i^T \mathbf{z}). \tag{58}$$

Since the logarithm is an increasing function, maximization of the likelihood is the same as maximization of the log-likelihood:

$$\log L(\mathbf{v}_1, \mathbf{v}_2, \ ..., \ \mathbf{v}_{n_1}) = |\det(\mathbf{V})| + \sum_{i=1}^{n_1} \log p(\mathbf{v}_i^T \mathbf{z}). \tag{59}$$

If we knew the function $p$ we could estimate the features $\mathbf{V}$ by maximizing the Eq. (59) with respect to $\mathbf{V}$. However, we do not really estimate the features $\mathbf{v}_i$, but rather learn them by some intuitively justified statistical criteria. That criteria is maximization of the sparseness of feature outputs $s_i$. Namely, function $\log p(s)$ is chosen to be such to measure the sparseness of feature outputs, i.e. maximization of the likelihood is equivalent to maximization of the sparsenesses of the outputs if the function $\log p(s)$ are of the form required for sparseness measurements, that is:

$$\log p(s) = h(s^2). \tag{60}$$

where $h$ is a convex function and $\mathbf{s} = \mathbf{V}\mathbf{z}$. In practice, $h$ is chosen to be the negative square root (more on the possible choices of $h$ in [8]). In that case log-likelihood takes the following form:

$$\log L(\mathbf{v}_1, \mathbf{v}_2, \ ..., \ \mathbf{v}_{n_1}) = |\det(\mathbf{V})| - \sqrt{\sum_{i=1}^{n_1} (\mathbf{v}_i^T \mathbf{z})^2}. \tag{61}$$

The main idea of topographic ICA is to model the topological arrangement of the features in the same way the cells in the visual cortex are arranged. It is known

that the cells in the visual cortex have a specific spatial organization. When moving on the cortical surface, the response properties of the neurons change in systematic way. This phenomenon is called topographic organization. To model topographic organization, we have to first define which features are close to each other on the cortical surface. This is done by arranging the features $s_i$ on a two-dimensional grid[8]. This is illustrated in Fig 15. The topography is formally expressed by a neighborhood function, i.e. pooling matrix $\pi(i, j)$ that described the proximity of the features (components) $\mathbf{v}_i$ and $\mathbf{v}_j$. This matrix is described in 3.1.

If we incorporate the idea of topographic ordering, instead of looking at the sparseness of the outputs $s_i$ we are looking at the sparseness of the outputs $c_j$ modeled as:

$$c_j = \sum_{i=1}^{n_1} \pi(i, j) s_i^2, \tag{62}$$

where $j = 1,\ ...,\ n_2$, $n_2$ being the number of outputs after the pooling operation. The $\log p$ function is then:

$$\log p(s) = h(c), \tag{63}$$

and the log-likelihood of topographic ICA is:

$$\log L(\mathbf{v}_1, \mathbf{v}_2,\ ...,\ \mathbf{v}_{n_1}) = \log |\det(\mathbf{V})| - \sum_{j=1}^{n_2} \sqrt{\sum_{i=1}^{n_1} \pi(i, j)(\mathbf{v}_i^T \mathbf{z})^2}. \tag{64}$$

Finally, we observe the log-likelihood of multiple image patches. As we can assume that the patches are independent from each other, the probability of observing $T$ patches is the product of the probabilities of observing each patch independently:

$$L(\mathbf{v}_1, \mathbf{v}_2,\ ...,\ \mathbf{v}_{n_1}) = \prod_{t=1}^{T} p(\mathbf{z}^t). \tag{65}$$

The log-likelihood of all patches is:

$$\log L(\mathbf{v}_1, \mathbf{v}_2, \ ..., \ \mathbf{v}_{n_1}) = T \log |\det(\mathbf{V})| - \sum_{t=1}^{T} \sum_{j=1}^{n_2} \sqrt{\sum_{i=1}^{n_1} \pi(i,j)(\mathbf{v}_i^T \mathbf{z}^t)^2}. \tag{66}$$

The topography given by $\pi(i, j)$ is considered fixed, and only the linear feature weights $\mathbf{v}_i$ need to be estimated, so this likelihood is a function of the $\mathbf{v}_i$ only. The vectors $\mathbf{v}_i$ are constrained to form an orthogonal matrix, so the determinant is constant (one) and the term $T \log |\det(V)|$ can be ignored. Therefore, the minimization of Eq. (2) follows directly.